

Münster 1969

A. Hechow

Tag der mündlichen Prüfung: 7. Februar 1969

Dekan: Prof. Dr. phil. Heinrich Dörrie

Referent: Prof. Dr. phil. Peter Hartmann

Korreferent: Prof. Dr. phil. Heinrich Lausberg



INHALT

1. Einführung	1
1.1 Zur Problemstellung	1
1.2 Grundlegende Begriffe der algebraischen Linguistik	3
1.3 Grundlegende Begriffe der Automatentheorie	6
2. MEALY-Automaten und MOORE-Automaten	10
2.1 Grundlegende Begriffe der Theorie der MEALY-Automaten	10
2.2 Sätze über MEALY-Automaten	14
2.3 MOORE-Automaten als spezielle MEALY-Automaten	18
3. Beschreibung von Sprachen	21
3.1 Zur Problemstellung	21
3.2 Die Darstellbarkeit aller Sprachen durch Automaten	23
3.3 Die Darstellbarkeit von Sprachen in Automaten durch Zustände	
4. Finite-State-Akzeptoren und -Sprachen	30
4.1 Determinierte und nichtdeterminierte Finite-State-Akzeptoren	30
4.2 Sätze über determinierte Finite-State-Akzeptoren	35
4.3 Finite-State-Sprachen	41
5. Zur Charakterisierung von Finite-State-sprachen	42
5.1 Das Analyse-Synthese-Theorem	42
5.2 Die Charakterisierungssätze von MYHILL und NERODE	50
6. Operationen über Finite-State-Sprachen	55
6.1 BOOLEsche Operationen	56
6.2 Substitution	56
6.3 Sequenzentransducer-Abbildungen	58
6.4 Ein Entscheidungsproblem	61
7. Umschreibesysteme	64
7.1 Grundlegende Begriffe der Theorie der kontextfreien Sprachen	65
7.2 Die Äquivalenz der Begriffe 'Finite-State-Akzeptor' und 'einseitig lineare Grammatik'	68

7.3 Folgerungen	76
8. Die Anwendung der Theorie auf das Italienische	78
8.1 Das Explizitmachen einer abgekürzten FSG	78
8.2 Die Generation italienischer Sätze	80
8.3 Die Analyse italienischer Sätze bzw. Nicht-Sätze	83
8.4 Die Umwandlung einer unbeschränkten FSG in eine determinierte FSG	85
8.5 Ein Algorithmus zur MOORE-Reduktion von DFAen	86
8.6 Zusammenfassung der Ergebnisse	87
Zur benutzten Literatur	89
Symbole und Abkürzungen	90
Sachverzeichnis	91
Errata	93

Der Dokumentationsteil zu dieser Untersuchung befindet sich in der Bibliothek des Instituts für Sprachwissenschaft der Universität Münster und des Fachbereichs Sprachwissenschaft der Universität Konstanz.

ZUR THEORIE DER FINITE-STATE-SPRACHEN MIT EINER ANWENDUNG AUF DAS ITALIENISCHE

1. EINFÜHRUNG

1.1 Zur Problemstellung

Zu den Aufgaben der Linguistik könnte man z.B. rechnen: eine Grammatik für eine bestimmte natürliche Sprache zu schreiben oder eine Theorie der Übersetzung einer bestimmten natürlichen Sprache in eine andere aufzustellen. Ferner könnte man in diesem Zusammenhang untersuchen, wie Grammatiken bzw. Übersetzungstheorien für natürliche Sprachen überhaupt beschaffen sein müssen (man käme dann zu einer Theorie der allgemeinen Grammatik bzw. der 'allgemeinen Übersetzung'). Auch sind etwa Untersuchungen zum Sprachvermögen denkbar.

Diese und viele andere Aufgaben sind in der traditionellen Linguistik in Angriff genommen worden; dabei sind die Fragestellungen oft nicht besonders präzise formuliert worden. Dennoch hat die Forschung zweifellos wichtige Entdeckungen gemacht. Allerdings ging die traditionelle Forschung meist intuitiv vor, d.h. die Richtigkeit von Ergebnissen lies sich meist nicht an Hand von strengen Methoden überprüfen. Die Forschungsergebnisse waren umso anfechtbarer, je kompliziertere Sachverhalte sie zum Gegenstand hatten. So wird man sich beispielsweise heute über die korrekte Formulierung von Lautgesetzen eher einigen können als etwa über die Richtigkeit der HUMBOLDT'schen Sprachtheorie. Im letzteren Fall hätte man wohl schon große Schwierigkeiten, überhaupt zu verstehen, was mit der HUMBOLDT'schen Terminologie genau gemeint ist.

In der Gegenwart findet in der Linguistik eine Reflexion auf die Methoden statt. Man schafft Beschreibungstheorien, die es erlauben, Ergebnisse in weitgehend überprüfbarer Form darzustellen. Im Idealfall kann eine Rechenmaschine diese Überprüfung leisten. Die selbstaufgelegte Strenge im Vorgehen hat allerdings zur Folge, daß man nur verhältnismäßig kleine oder gar 'triviale' Bereiche der traditionellen Linguistik exakt beschreiben kann. (So weiß man z.B. in unserem Fall schon vor Beginn der Untersuchung, daß man mit den Methoden der Theorie der Finite-State-Sprachen nicht alle Phänomene von natürlichen Sprachen beschreiben kann.)

Man sollte dies aber nicht als einen Mangel ansehen. Der Bereich der sogenannten Geisteswissenschaften ist eben derartig kompliziert strukturiert, daß er (nach Meinung des Verfassers) erst heutzutage der exakten Forschung - durch Entwicklung neuer Deskriptionsmethoden - zugänglich gemacht wird. Erst wenn die Fragestellungen genügend präzisiert sind und wenn einfache Phänomene dieses Bereiches exakt beschrieben sind, kann man hoffen, kompliziertere Sachverhalte in den Griff zu bekommen. Wie kompliziert etwa natürliche Sprachen sind, kann man daran ermessen, daß die im Anwendungsteil benutzte generative Grammatik 8000 Regeln enthält, obwohl sie nur einen - nicht einmal besonders komplizierten-Satztyp des Italienischen beschreibt. Das besagt natürlich nicht, daß dieselbe Teilsprache des Italienischen in einer anderen Theorie, als der hier benutzten, nicht wesentlich einfacher zu beschreiben wäre.

Wichtigstes Ziel der vorliegenden Arbeit ist es, die Möglichkeit der Anwendung der Theorie der Finite-State-Sprachen an Hand eines Beispiels (des Italienischen) zu untersuchen. In extenso ist dies, soweit dem Verfasser bekannt, noch nicht geleistet worden.

Den theoretischen Teil der Arbeit haben wir möglichst allgemein aufgebaut, um die Theorie auch für andere Fragestellungen, als die hier untersuchte, benutzen zu können; die spezielle Themenstellung der Arbeit wurde allerdings stets im Auge behalten. Die Berechtigung dieses Vorgehens scheint nach Abschluß der Untersuchung u.a. dadurch bestätigt zu werden, daß sich ergeben hat, daß die Theorie der Finite-State-Sprachen für die Beschreibung natürlicher Sprachen noch ungeeigneter ist, als allgemein vermutet wird (BAR HILLEL (1964,93) etwa sagt: The use of FSGs (= Finite-State-Grammars) as models for natural languages has many attractive features.), ihr dagegen für die Behandlung der 'niedrigen' Ebenen der Sprachbeschreibung große Bedeutung zukommen dürfte; dies trifft etwa zu für Probleme der Beschreibung von Lautgesetzen der historischen Grammatik oder von phonetischen, phonematischen oder graphematischen Sachverhalten. Für die Behandlung derartiger Fragen können Teile der hier dargestellten Theorie übernommen werden. Ferner stellen sich die hier behandelten Probleme in abgewandelter Form auch in komplizierteren Theorien (man denke etwa an Charakterisierungssätze, an BOOLEsche Operationen, an die Substitution, Sequenzentransducer-Abbildungen, an die

Äquivalenz von Umschreibesystemen mit anderen formalen Systemen oder an Entscheidungsprobleme); die Arbeit gibt so implizit einen gewissen Einblick in die in der algebraischen Linguistik benutzten Methoden.

Der Gang der Untersuchung ist folgender: zuerst werden Aussagen über die Leistungsfähigkeit der Automatentheorie im allgemeinen (Kapitel 1 bis 3) gemacht; die Kapitel 4 bis 7 beschäftigen sich mit der eigentlichen Theorie der Finite-State-Sprachen. In Kapitel 8 wird dann die Anwendbarkeit der Theorie diskutiert.

1.2 Grundlegende Begriffe der algebraischen Linguistik

Def.1.2.1 Ein Vokabular (Alphabet) ist eine nichtleere Menge identifizierbarer, paarweise unterscheidbarer Zeichen.

Aus den Elementen von V können Ketten gebildet werden (Ketten über V), u.a. auch die leere Kette, die aus keinem Element von V besteht.

Def.1.2.2 Induktive Def. der Ketten(über V) und deren Länge

- 1) Die leere Kette e ist eine Kette über V ; sie hat die Länge $l(e)=0$.
- 2) Wenn a ein Element von V und p eine Kette über V ist mit $l(p)=n$, so sind ap und pa Ketten über V mit $l(ap)=l(pa)=n+1$.

Für jede Kette p ist $cp=p=pe$.

Alle Ketten im Sinne dieser Def. haben eine endliche Länge.

Def.1.2.3 Das freie Monoid über V ($F(V)$) (die freie Halbgruppe über V) ist die Menge aller Ketten über V .

Das freie Monoid über V enthält abzählbar unendlich viele Ketten.

Def.1.2.4 Jede Teilmenge L_V von $F(V)$ heißt Sprache (über V).

Der Begriff Sprache wird hier in einem allgemeineren Sinne als in der Umgangssprache benutzt. Da es sich hier nur um eine Menge von Ketten handelt - es könnten etwa auch Glasketten sein oder die Knotenschnüre der Inkas - wäre ein terminus technicus angebracht. So spricht man in der Automatentheorie oft von Ereignissen, wenn Sprachen im algebraischen Sinne gemeint sind. Da sich der Terminus Sprache aber in der Literatur zur algebraischen Linguistik eingebürgert hat,

wie er oben definiert ist, werden wir im folgenden natürliche Sprache sagen, wenn vom Sprache im üblichen Sinn die Rede ist.

Def.1.2.5 Eine Kette, die Element einer Sprache L ist, wird Satz von L genannt.

Def.1.2.6 Eine generative Grammatik (über V) G der Sprache L (über V) ist ein finites Verfahren zur Aufzählung (Konstruktion, Erzeugung) aller Sätze von L . G heißt auch Aufzählungsverfahren. Eine durch eine generative Grammatik gegebene Sprache L nennen wir auch $L(G)$. Eine derartige Sprache heißt aufzählbar.

Def.1.2.7 Eine Erkennungsgrammatik G (Analysegrammatik) für die Sprache L_V ist ein finites Verfahren, das für beliebige Ketten p über V in endlich vielen Schritten zu entscheiden erlaubt, ob p ein Satz von L ist oder nicht. G heißt auch Entscheidungsverfahren. Eine Sprache, zu der es eine Erkennungsgrammatik gibt, heißt entscheidbar.

Ohne Beweis sei hier bemerkt, daß es nicht für alle Sprachen, die durch eine generative Grammatik gegeben sind, eine Erkennungsgrammatik gibt. Ein Beispiel dafür ist die sog. Prädikatenlogik der ersten Stufe. In diesem Sinne ist es u.U. unproblematischer, Sprachen nur unter dem Aspekt der Erzeugung (Generation, Produktion) - wie dies in der Literatur meist geschieht - zu behandeln, als vielmehr auch den Aspekt der Analyse zu berücksichtigen.

Die in der Literatur zur sog. "Generativen Grammatik" betrachteten generativen Grammatiken sind oft zugleich Analysegrammatiken. Dies gilt jedoch nicht allgemein; besonders sind die sog. unbeschränkten Transformationsgrammatiken im allgemeinen keine Entscheidungsverfahren.⁺ Wie schon gesagt ist nun zwar nicht jede aufzählbare Sprache entscheidbar, aber für natürliche Sprachen kann man vernünftigerweise die Entscheidbarkeit fordern. Deswegen ist es sinnvoll, natürliche Sprachen

⁺ Die in der Literatur zur Transformationsgrammatik eingeführten Restriktionen haben meist die Funktion, die durch sie eingeschränkten Transformationsgrammatiken zur Analyse benutzen zu können.

durch Analysegrammatiken zu beschreiben und sich nicht mit einer generativen Grammatik zufrieden zu geben, die u.U. kein Entscheidungsverfahren ist.

Andererseits kann man aus einer Analysegrammatik sofort eine Erzeugungsgrammatik konstruieren, und zwar folgendermaßen.⁺⁾ Sei A eine Analysegrammatik mit $L=L(A) \subseteq F(V)$. Gesucht ist eine generative Grammatik G mit $L(G)=L=L(A)$. G wollen wir als eine berechenbare Funktion verwirklichen, deren Definitionsbereich der Bereich der natürlichen Zahlen ist. Für den Wertebereich L von G ist dann eine natürliche Reihenfolge $G(0), G(1), G(2), \dots$ festgelegt, in der L , eventuell mit Wiederholungen, durchlaufen wird. Wir sagen dann, daß G die Sätze von L in der Reihenfolge $G(0), G(1), G(2), \dots$ aufzählt.

Nun wollen wir die Funktion G konstruieren. Zunächst gilt: da es zu jeder natürlichen Zahl n nur endlich viele Ketten der Länge n gibt, ist $F(V)$ aufzählbar, d.h. es gibt eine berechenbare Funktion, die auf den natürlichen Zahlen definiert ist und $F(V)$ aufzählt; sei f diese Funktion. Nun definieren wir für alle $p \in F(V)$:

$$g(p) := \begin{cases} p, & \text{falls } p \in L \\ \text{ein festes } q \in L & \text{sonst} \end{cases} .$$

g ist offensichtlich berechenbar, denn L ist entscheidbar. Deshalb können wir für alle natürlichen Zahlen n definieren:

$$G(n) := g(f(n)) .$$

Weil f und g berechenbar sind, ist auch G berechenbar, und offensichtlich zählt G die Sätze von L auf.

G ist also eine generative Grammatik für L im Sinne Def.1.2.6. (Allerdings wird der Terminus "generative Grammatik" hier in einem allgemeineren Sinn gebraucht, als es in der Literatur üblich ist.) Man hat also mit einer Analysegrammatik für eine Sprache L zugleich eine Erzeugungsgrammatik L , wenn man die dargelegte Konstruktion anwendet. Die Funktion f muß dazu natürlich effektiv gegeben werden.

⁺⁾ cf. für das folgende HERMES, 11.

1.3 Grundlegende Begriffe der Automatentheorie

Zur Aufzählung und Analyse einer Sprache gibt es ~~verschie-~~dene Arten von formalen Systemen, z.B. POST-Kalküle, rekursive Funktionen oder Turingmaschinen. In der algebraischen Linguistik sind bisher die sog. Umschreibesysteme oder Ersetzungssysteme (rewriting systems) bevorzugt untersucht worden. Dies vielleicht deshalb, weil sie es erlauben, den traditionellen Begriff der Struktur besonders anschaulich zu präzisieren, und zwar durch sog. "Bäume". Die formalen Systeme, die hier untersucht werden sollen, liefern zwar keine echten Bäume ⁺), dafür aber eine einheitliche Theorie der Grammatik und der Übersetzung. Diese beiden großen Forschungsbereiche der Linguistik stehen hier in einem derartigen Zusammenhang, daß man die Theorie der Grammatik als einen Spezialfall der Theorie der Übersetzung ansehen kann.

Wir führen nun zunächst ein sehr allgemeines Übersetzungskonzept ein, das zusammen mit den folgenden Begriffen zum theoretischen Apparat der Automatentheorie gehört.

Def.1.3.1 Ein (unbeschränkter) Transducer (TR) $T=(S,X,Y,R)$ ist ein 4-Tupel, wobei

- 1) S eine beliebige Menge (die Zustände von T),
- 2) X ein endliches Vokabular (das Eingabealphabet von T),
- 3) Y ein endliches Vokabular (das Ausgabealphabet von T),
- 4) R eine beliebige Teilmenge von $S \times F(X) \times F(Y) \times S$ ⁺⁺)
(die Regeln von T) ist.

Elemente des Eingabe- bzw. Ausgabealphabets von T nennen wir Eingabe- bzw. Ausgabesymbole.

Die Arbeitsweise eines TRs ist durch seine Regeln festgelegt. Eine Regel (s,p,q,t) besagt: wenn sich der TR im Zustand s befindet und die Eingabekette p liest, dann gibt er die Kette q aus und geht in den Zustand t über. Wir erläutern im nächsten Kapitel die Arbeitsweise von TRn einer speziellen Klasse genauer. ⁺⁺⁺⁾

⁺) Die sog. einseitig linearen Grammatiken und die Finite-State-Grammatiken liefern nur degenerierte Bäume. Kontextfreie Grammatiken, die echte Bäume liefern, werden zwar eingeführt, aber hier nicht weiter untersucht; cf. 7.1.

⁺⁺) $M_1 \times \dots \times M_n$ ($2 \leq n$) bezeichnet das cartesische Produkt der Mengen M_1 bis M_n , d.h. die Menge der n-Tupel $(x_1, \dots, x_n)$ mit $x_i \in M_i$ ($1 \leq i \leq n$).

⁺⁺⁺) cf. 2.1

Def.1.3.2 Sei $T=(S,X,Y,R)$ ein TR, s ein beliebiger Zustand von T ; dann gelte für alle $p \in F(X)$:

$$T_s(p) := \left\{ q \in F(Y) : \begin{array}{l} \text{es gibt } p_1, \dots, p_k \in F(X), \\ q_1, \dots, q_k \in F(Y), \\ t_1, \dots, t_{k+1} \in S \text{ mit:} \end{array} \right.$$

- 1) $p = p_1 \dots p_k$,
- 2) $q = q_1 \dots q_k$,
- 3) $(t_i, p_i, q_i, t_{i+1}) \in R (1 \leq i \leq k)$
und
- 4) $t_1 = s$ } +)

Nun definieren wir für alle $L \subseteq F(X)$:

$$T_s^*(L) := \bigcup_{p \in L} T_s(p).$$

$T_s(p)$ heißt die Menge der durch s von T festgelegten Übersetzungen von p ($T_s(p)$ ist u.U. leer). T_s ist offenbar eine eindeutige Abbildung von $F(X)$ in $P(F(Y))$. T_s^* dagegen ist eine eindeutige Abbildung von $P(F(X))$ in $P(F(Y))$.⁺⁺⁾

T_s nennen wir die durch s von T festgelegte Transducerabbildung.

Es ist klar, daß man mit einem unbeschränkten TR T jede denkbare Übersetzung 'festlegen' kann, da die Zustands- und Regelmenge von T keinen Einschränkungen unterliegen; so können R oder S z.B. sogar überabzählbar unendlich sein. Aus diesem Grund ist ein unbeschränkter TR i.a. kein finites Verfahren; nur finite Verfahren sind aber praktisch realisierbar. Wir wollen daher für TR Einschränkungen formulieren, von denen insbesondere die sog. Endlichkeitsbedingung schließlich zu finiten Verfahren führt.⁺⁺⁺⁾

^{+) Die p_i, q_i, t_i sind hier Variablen für Elemente der entsprechenden Mengen. Auch im folgenden wird, um die Notation nicht zu sehr zu belasten, auf die genaue Kennzeichnung von Variablen im Unterschied zu Konstanten verzichtet, wenn aus dem Zusammenhang klar ist, was gemeint ist.}

^{++) $P(M)$ bezeichnet die Potenzmenge von M , d.h. die Menge aller Teilmengen der Menge M .}

⁺⁺⁺⁾ cf. Def. 1.3.5

Eine derartige mögliche Einschränkung ist die Determinierung von TR_n .

Def.1.3.3 Ein $TR\ T=(S,X,Y,R)$ heißt determiniert, wenn

- 1) Es für jedes $s \in S$, $p \in F(X)$ höchstens eine Regel (s,p,q,t) gibt und
- 2) für alle $s \in S$ gilt: $(s,e,e,s) \in R$.

In einem determinierten TR gibt es so insbesondere zu jedem $(s,x) \in S \times X$ höchstens ein $y \in Y$ bzw. $t \in S$; man beachte dies im nächsten Kapitel.

Die Regeln eines TR s können nun in bezug auf die zulässige Eingabe oder Ausgabe eingeschränkt werden.

Def.1.3.4 Ein $TR\ T=(S,X,Y,R)$ hat

- 1) eine sequentielle Eingabe und Ausgabe, wenn $R \subseteq S \times F(X) \times F(Y) \times S$,
- 2) eine sequentielle Eingabe und Buchstabenausgabe, wenn $R \subseteq S \times F(X) \times Y \times S$,
- 3) eine Buchstabeneingabe und sequentielle Ausgabe, wenn $R \subseteq S \times X \times F(Y) \times S$ und
- 4) eine Buchstabeneingabe und -ausgabe, wenn $R \subseteq S \times X \times Y \times S$.

Schließlich kann man TR noch in bezug auf die Anzahl der Zustände oder Regeln beschränken. Wir betrachten unter den möglichen Einschränkungen nur die folgende:

Def.1.3.5 (Endlichkeitsbedingung für TR) Ein $TR\ T=(S,X,Y,R)$ heißt endlich, wenn S und R endliche Mengen sind.

Einige in der Literatur zur Automatentheorie wichtige Typen von TR_n seien hier genannt:

- 1) die determinierten TR mit Buchstabeneingabe und -ausgabe (MEALY- und MOORE-Automaten), die in Kapitel 2 behandelt werden;
- 2) die endlichen TR mit sequentieller Eingabe und Ausgabe (Sequential Transducer), die in 6.3 behandelt werden;
- 3) die endlichen determinierten TR mit Buchstabeneingabe und sequentieller Ausgabe (Generalized Sequential Machines);
- 4) die endlichen determinierten TR mit Buchstabeneingabe und -ausgabe (endliche Automaten).

Von den TRn kann man (indem man die Ausgabe fortläßt) in naheliegender Weise zu den Akzeptoren übergehen, die man, falls sie finit gegeben sind, als Grammatiken ansehen kann.

Def.1.3.6 Ein (unbeschränkter) Akzeptor $A=(S,V,R)$

ist ein Tripel, wobei

- 1) S eine beliebige Menge ist (die Zustände von A),
- 2) V ein endliches Vokabular ist (das Alphabet von A) und
- 3) R eine beliebige Teilmenge von $S \times F(V) \times S$ ist (die Regeln von A).

Wie man mit Hilfe von TRn Abbildungen definieren kann, so kann man durch Akzeptoren Sprachen festlegen.

Def.1.3.7 Sei $A=(S,V,R)$ ein Akzeptor, $I, F \subseteq S$; dann definieren wir:

$L_I^F := \{ p \in F(V) : \text{es gibt } p_1, \dots, p_k \in F(V) \text{ und } t_1, \dots, t_{k+1} \in S, \text{ so daß}$

gilt:

- 1) $p = p_1 \dots p_k$,
- 2) $(t_i, p_i, t_{i+1}) \in R \ (1 \leq i \leq k)$,
- 3) $t_1 \in I$
und
- 4) $t_{k+1} \in F$.

L_I^F heißt die durch die Zustandsmengen I und F von A festgelegte Sprache (über V).

Für Akzeptoren kann man analoge Einschränkungen wie für TR formulieren. Entsprechend könnte man die Leistungsfähigkeit von TRn verschiedener Typen (d.h. von TRn, die verschiedenen Einschränkungen unterworfen sind) untersuchen. Dies wollen wir hier nicht tun, denn das Ziel unserer Arbeit ist lediglich die Untersuchung von endlichen TRn und besonders von endlichen Akzeptoren.^{+))}

Bevor wir aber dazu übergehen, endliche TR und Akzeptoren sowie die ihnen entsprechenden Grammatiken zu untersuchen, wollen wir noch auf die Leistungsfähigkeit von TRn einer besonders wichtigen Klasse, die sog. MEALY- und MOORE-Automaten eingehen. Es wird sich zeigen, daß die eingeführten TR letztlich noch genauso allgemein wie die unbeschränkten TR sind, sie also für praktische Zwecke nicht brauchbar sind.⁺⁺⁾ Wir gewinnen aber im nächsten Kapitel

^{+))} cf. 4.1. ⁺⁺⁾ cf. p.7

wichtige allgemeine Einsichten in das Funktionieren von TRn, die wir für den Bereich der endlichen TR und Akzeptoren verwerten können. Explizit werden wir in diesem allgemeinen Teil nur solche Sätze beweisen, die sich direkt auf endliche TR und Akzeptoren übertragen lassen.

2. MEALY-AUTOMATEN UND MOORE-AUTOMATEN

2.1 Grundlegende Begriffe der Theorie der MEALY-Automaten

Def.2.1.1 Ein MEALY-Automat $A=(S,X,Y,n,o)$ ist ein 5-Tupel, wobei

- 1) S eine beliebige (nichtleere) Menge ist (die Zustände von A),
 - 2) X ein endliches Vokabular (das Eingabealphabet von A),
 - 3) Y ein endliches Vokabular (das Ausgabealphabet von A),
 - 4) n eine eindeutige Abbildung der Menge $S \times X$ in die Menge S ist (die Überföhrungsfunktion von A ; sie überföhrt den Automaten unter dem Einfluß eines Eingabesymbols in einen neuen Zustand.)
- und
- 5) o eine eindeutige Abbildung der Menge $S \times X$ in Y ist (die Output-Funktion oder Ergebnisfunktion von A).

Ist außerdem noch ein Element s_0 von S als Anfangszustand von A ausgezeichnet, so wird A initialer Automat genannt.

Jeder MEALY-Automat ist offenbar ein determinierter TR mit Buchstabeneingabe und -ausgabe, dessen Regeln sich schreiben lassen als $(s, x, n(s, x), o(s, x))$ für beliebige Zustände s und Eingabesymbole x .⁺⁾ Statt 'MEALY-Automat' sagen wir auch kurz 'Automat'. Automaten mit endlicher Zustandsmenge heißen endlich.⁺⁺⁾

⁺⁾ cf. Def.1.3.3

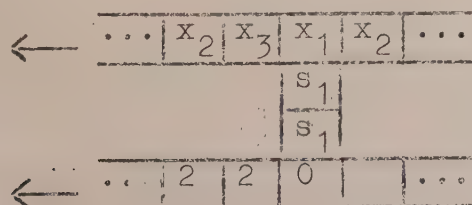
⁺⁺⁾cf. Def.1.3.5

Wir wollen uns die Arbeitsweise eines endlichen Automaten an Hand eines Beispiels klarmachen. Dabei benutzen wir zur Definition von n und o eine (n,o) -Matrix, die für einen Automaten mit der Zustandsmenge $\{s_1, \dots, s_k\}$, dem Eingabealphabet $\{x_1, \dots, x_n\}$ und dem Ausgabealphabet $\{o(s_i, x_j) : 1 \leq i \leq k, 1 \leq j \leq n\}$ folgendermaßen aussieht:

	x_1	...	x_j	...	x_n
s_1	$n(s_1, x_1)o(s_1, x_1)$	...		...	$n(s_1, x_n)o(s_1, x_n)$
$\vdots$		...		...	
s_i		...	$n(s_i, x_j)o(s_i, x_j)$	...	
$\vdots$		...		...	
s_k	$n(s_k, x_1)o(s_k, x_1)$	...		...	$n(s_k, x_n)o(s_k, x_n)$

In jedem Feld dieser Matrix steht eine n -Eintragung und eine o -Eintragung. Der Automat befindet sich jeweils in einem Zustand s_j ($1 \leq j \leq k$). Dabei liest er das Symbol x_k ($1 \leq k \leq n$) des Eingabealphabetes. Er geht dann in den neuen Zustand $n(s_j, x_k)$ über und gibt dabei das Zeichen $o(s_j, x_k)$ aus. Offenbar werden die möglichen Verhaltensweisen eines endlichen Automaten durch seine (n,o) -Matrix zureichend beschrieben.

Einen Automaten kann man sich gut als einen Mechanismus mit einem Eingabe- und Ausgabeband vorstellen, der Ketten vom Eingabeband, das sich von rechts nach links bewegt, Symbol für Symbol übersetzt und die Ergebnisse auf das Ausgabeband schreibt, daß sich in gleicher Richtung und mit gleicher Geschwindigkeit bewegt. Graphisch stellt man Automaten in der Literatur als zwei Bänder dar, zwischen denen sich die 'Kontrolle' befindet, die jeweils ein Feld des Eingabebandes liest und auf ein Feld des Ausgabebandes ein Symbol druckt. Dabei kann man auf die Kontrolle den Zustand, in dem sich der Automat befindet, schreiben und den Zustand, in den er als nächstes übergeht. So stellt das Bild



einen Automaten dar, der sich im Zustand s_1 befindet, x_1 liest, 0 druckt⁺ und in den Zustand s_1 übergeht. Die Richtung, in der sich die Bänder bewegen, ist durch die Pfeile angedeutet.

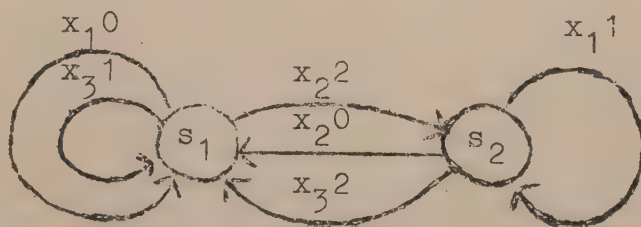
Sei etwa $A=(S,X,Y,n,o)$ ein Automat mit $s=\{s_1,s_2\}$, $X=\{x_1,x_2,x_3\}$, $Y=\{0,1,2\}$, und seien n und o durch die folgende (n,o) -Matrix definiert:

	x_1	x_2	x_3
s_1	$s_1 0$	$s_2 2$	$s_1 1$
s_2	$s_2 1$	$s_1 0$	$s_1 2$

A befinde sich zu Beginn der Übersetzung im Zustand s_1 und lese das erste Symbol der Kette $p=x_2x_3x_1x_2$; p wird in die Kette 2202 übersetzt und nach getaner Arbeit befindet sich A im Zustand s_2 . Das Verhalten von A kann man sich durch das folgende Diagramm veranschaulichen:

A befindet sich in: $s_1 \ s_2 \ s_1 \ s_1 \ s_2$
A liest: $x_2 \ x_3 \ x_1 \ x_2$
A gibt aus: $2 \ 2 \ 0 \ 2$
A geht über in: $s_2 \ s_1 \ s_1 \ s_2$

A läßt sich auch als gerichteter Graph darstellen. Die Zustände von A werden dabei in Kreise geschrieben, die untereinander durch Pfeile verbunden sein können. Die Pfeile sind mit einem Eingabsymbol beschriftet und dem Ausgabesymbol, das der Automat beim Übergang von einem Zustand in einen anderen ausgibt. Für unser Beispiel sähe der Graph folgendermaßen aus:



Nun führen wir noch einige Konventionen ein. Das freie Monoid über dem Eingabealphabet eines Automaten (bzw. über dem Ausgabealphabet) nennen wir Eingabemonoid (bzw. Ausgabemonoid). Elemente dieser Monoide nennen wir Eingabewörter bzw. Ausgabewörter (des Automaten). Wenn wir Eingabewörter oder Ausgabewörter explizit als Ketten $x_1 \dots x_k$ hinschreiben, so sollen die x_i ($1 \leq i \leq k$) immer Variablen für Elemente des entsprechenden Vokabulars sein, wenn es nicht anders gesagt wird.
⁺) 'druckt' wird synonym zu 'gibt aus' benutzt.

Als nächstes verallgemeinern wir die Definitionen von n und o in naheliegender Weise:

Def.2.1.2 Sei A ein Automat, s ein beliebiger Zustand von A und $p=x_1 \dots x_k$ ein beliebiges Eingabewort; dann sei

$$N(s,p) := s_1 \dots s_k, \text{ wobei}$$
$$s_1 = n(s, x_1), s_2 = n(s_1, x_2), \dots, s_k = n(s_{k-1}, x_k)$$

und

$$O(s,p) := y_1 \dots y_k, \text{ wobei}$$
$$y_1 = o(s, x_1), y_2 = o(s_1, x_2), \dots, y_k = o(s_{k-1}, x_k).$$

Ferner sei für alle s von A

$$N(s,e) := s \text{ und } O(s,e) := e.$$

Offensichtlich sind N und O eindeutige Abbildungen von $S \times F(X)$ in $F(S)$ bzw. $F(Y)$. N heißt erweiterte Überföhrungs-funktion, O erweiterte Output-Funktion (Ergebnisfunktion) von A .

Def.2.1.3 Seien $A=(S_A, X, Y, n_A, o_A)$ und $B=(S_B, X, Y, n_B, o_B)$ beliebige Automaten mit gleichem Eingabe- und Ausgabealphabet. Unter einem Homomorphismus von A auf B versteht man eine eindeutige Abbildung f von S_A auf S_B , wobei gilt:

- 1) $f(n_A(s, x)) = n_B(f(s), x)$ und
- 2) $o_A(s, x) = o_B(f(s), x)$ für beliebige $s \in S_A, x \in X$.
- 3) Falls A und B initiale Automaten sind, dann muß f den Anfangszustand von A auf den Anfangszustand von B übertragen.

Ein Homomorphismus von A auf B , bei dem f umkehrbar eindeutig ist, heißt Isomorphismus von A auf B . Automaten A und B , für die es einen Homomorphismus⁺ von A auf B gibt, heißen homomorph (isomorph).

Man benötigt die Begriffe der Homomorphie und Isomorphie, um Sätze über Automaten zu beweisen, die verschieden definiert sind, aber (in einem noch näher zu bestimmenden Sinn) dasselbe leisten.

⁺) '(Isomorphismus)' einfügen!

Def.2.1.4 Sei $A=(S,X,Y,n,o)$ ein beliebiger Automat. Dann nennen wir die Menge aller paarweise verschiedenen Abbildungen A_s , wobei s ein bel. Zustand von A ist, die Abbildungsfamilie des Automaten A . Falls A ein initialer Automat mit dem Anfangszustand s_0 ist, so heit A_{s_0} die von A erzeugte Abbildung.

Offenbar ist fr jedes $p \in F(X)$ $A_s(p) = \{O(s,p)\}$ fr bel. $s \in S$.

Def.2.1.4 ist ein Spezialfall von Def.1.3.2. A_s ist die durch s von A festgelegte Transducerabbildung, die wir als Abbildung von $F(X)$ in $F(Y)$ auffassen knnen, wenn wir ^{+) Elemente des Bildbereiches, die smtlich Einermengen sind, mit den Elementen identifizieren, die sie enthalten; cf.p.7.}

Def.2.1.5 Automaten A und B mit gleichem Eingabe- und Ausgabealphabet X bzw. Y heien quivalent ($\dot{a}q$), wenn sie die gleiche Abbildungsfamilie festlegen. Initiale Automaten heien initialquivalent, wenn ihre Anfangszustnde die gleiche Abbildung erzeugen. Ferner heien zwei Zustnde eines (oder mehrerer) Automaten (mit gleichem Eingabe- und Ausgabealphabet) quivalent ($\dot{a}q$), wenn sie die gleiche Abbildung erzeugen.

Fr bel. Zustnde s von ⁺⁺⁾ A und t von B gilt also: $s \dot{a}q t$ g.d.w. $A_s(p) = B_t(p)$ fr bel. Eingabeketten p der Automaten. Dies ist offenbar genau dann der Fall, wenn $O_A(s,p) = O_B(s,p)$. $\dot{a}q$ ist ferner offensichtlich reflexiv, transitiv und symmetrisch, d.h. eine quivalenzrelation.

2.2 Stze ber MEALY-Automaten

Mit Hilfe der bisher eingefhrten Begriffe knnen wir nun einige Stze ber die quivalenz von Automaten formulieren. Aus Def.2.1.5 folgt sofort:

Satz 2.2.1 Automaten mit gleichem Eingabe- und Ausgabealphabet sind genau dann quivalent, wenn es fr jeden Zustand von A einen quivalenten Zustand von B und fr jeden Zustand von B einen quivalenten Zustand von A gibt.

Die Definition der quivalenz von Automaten kann als Przisierung der Bemerkung von p.13 aufgefat werden, da man Aussagen ber Automaten machen mchte, die dasselbe leisten, aber verschieden definiert sind. Fat man Automaten als Mechanismen auf, die Wrter in andere umschreiben, so ist klar, da quivalente Automaten die gleiche

^{+) 'die' einfgen}

^{++) 'A' einfgen}

Umschreibearbeit verrichten können; in äquivalenten Zuständen schreiben sie nämlich ein beliebiges Eingabewort in gleicher Weise um.

Satz 2.2.2 Sei f ein Homomorphismus des Automaten

$A=(S_A, X, Y, n_A, o_A)$ auf den Automaten $B=(S_B, X, Y, n_B, o_B)$.
Dann gilt für bel. $s \in S_A$: $s \dot{\sim} f(s)$, und daraus folgt nach Satz 2.2.1 sofort: $A \dot{\sim} B$.

Den Beweis findet man bei GLUSCHKOW, p.13.

Unser nächstes Ziel ist der Beweis von

Theorem 2.2.1 (Reduktionstheorem für MEALY-Automaten)

Vor.: Sei $\bar{A}$ die Klasse aller zu einem Automaten $A=(S_A, X, Y, n_A, o_A)$ äquivalenten Automaten.

Beh.: Es gibt einen Automaten B , so daß gilt:

- 1) $A \dot{\sim} B$.
- 2) Die Zustände von B sind paarweise inäquivalent.
- 3) Für jeden Automaten C in $\bar{A}$ gibt es einen Homomorphismus von C auf B .
- 4) In $\bar{A}$ gibt es keinen Automaten mit weniger Zuständen als B .

Zum Beweis benötigen wir

Lemma 2.2.1 Seien $A=(S_A, X, Y, n_A, o_A)$, $B=(S_B, X, Y, n_B, o_B)$
bel. Automaten (eventuell: $A=B$); dann
gilt für bel. $s \in S_A$, $t \in S_B$: wenn $s \dot{\sim} t$,
so $n_A(s, x) \dot{\sim} n_B(t, x)$ für bel. $x \in X$.

Beweis des Lemmas: Es genügt zu zeigen:

$$o_A(n_A(s, x), p) = o_B(n_B(t, x), p) \text{ für bel. } p \in F(X). \quad +)$$

Weil $s \dot{\sim} t$, so gilt zunächst:

$$o_A(s, xp) = o_B(t, xp) \text{ für bel. } p \in F(X); \text{ nun ist aber offenbar}$$

$$o_A(s, xp) = o_A(s, x) o_A(n_A(s, x), p) \quad ++)$$
 und

$$o_B(t, xp) = o_B(t, x) o_B(n_B(t, x), p) \text{ für bel. } p \in F(X), \text{ also}$$

$$o_A(s, x) o_A(n_A(s, x), p) = o_B(t, x) o_B(n_B(t, x), p);$$

weil $s \dot{\sim} t$, so $o_A(s, x) = o_B(t, x)$ und folglich

$$o_A(n_A(s, x), p) = o_B(n_B(t, x), p) \text{ für bel. } p \in F(X).$$

+) cf. die Bemerkung nach Def.2.1.5, p.14

++) Das folgt sofort aus Def.2.1.2, p.13.

Beweis von Theorem 2.2.1

Sei $A=(S_A, X, Y, n_A, o_A)$ ein bel. Automat. Wir geben einen Automaten B an, für den wir die oben formulierten Behauptungen 1 bis 4 zeigen.

$B=(S_B, X, Y, n_B, o_B)$ wird wie folgt konstruiert. Wir zerlegen zunächst S_A in die Teilmengen der zueinander äquivalenten Zustände. Diese Teilmengen sind offenbar disjunkt und heißen die Äquivalenzklassen von A ; wir schreiben $\bar{s} := \{t: t \text{ äq } s\}$ für bel. $s \in S_A$. Sei $S_B := \{\bar{s}: s \in S_A\}$, d.h. S_B ist die Menge der Äquivalenzklassen von A .

Nun definieren wir für bel. $\bar{s} \in S_B, x \in X$:

$$o_B(\bar{s}, x) := o_A(t, x),$$

wobei t ein bel. Element der Äquivalenzklasse $\bar{s}$ ist. $o_B(\bar{s}, x)$ ist unabhängig vom Repräsentanten definiert, denn für äquivalente Zustände ist die Ergebnisfunktion für bel. $x \in X$ gleich definiert.

Sei nun für alle $\bar{s} \in S_B, x \in X$:

$$n_B(\bar{s}, x) := \bar{t}, \text{ wobei } \bar{t} = \overline{n_A(r, x)}$$

und r ein bel. Element von $\bar{s}$ ist. Diese Definition ist eindeutig, denn nach Lemma 2.2.1 werden äquivalente Zustände durch n_A wieder in äquivalente Zustände überführt.

Beh.1: $A \text{ äq } B$.

Wir definieren für bel. $s \in S_A$: $f(s) := \bar{s}$.

Wir zeigen, daß für alle $s \in S_A$: $s \text{ äq } f(s)$. Damit ist dann Beh.1 gezeigt, denn f ist offensichtlich eine eindeutige Abbildung von S_A auf S_B , und so gibt es auch für jeden Zustand von S_B einen äquivalenten Zustand von S_A und folglich sind nach Satz 2.2.1 A und B äquivalent.

Sei nun s ein bel. Element von S_A , $p = x_1 \dots x_k$ ein bel. Eingabewort. Es sei

$$N_A(s, p) = s_1 \dots s_k \text{ und } O_A(s, p) = y_1 \dots y_k,$$

$$N_B(\bar{s}, p) = \bar{r}_1 \dots \bar{r}_k \text{ mit } \bar{r}_1 = n_B(\bar{s}, x_1), \dots, \bar{r}_k = n_B(\bar{r}_{k-1}, x_k) \text{ und}$$

$$O_B(\bar{s}, p) = q_1 \dots q_k \text{ mit } q_1 = o_B(\bar{s}, x_1), \dots, q_k = o_B(\bar{r}_{k-1}, x_k).$$

Wir müssen zeigen: $y_1 \dots y_k = q_1 \dots q_k$; dann gilt: $s \text{ äq } f(s)$ und Beh.1 ist gezeigt.⁺

⁺) cf. die Bemerkung nach Def.2.1.5, p.14

Dann gilt:

1. $\bar{r}_1 = n_B(\bar{s}, x_1) = \overline{n_A(s, x_1)} = f(n_A(s, x_1)) = f(s_1) = \bar{s}_1$
2. $\bar{r}_2 = n_B(\bar{r}_1, x_2) \underset{1. \text{ Zeile}}{=} n_B(\bar{s}_1, x_2) = \overline{n_A(s_1, x_2)} = f(n_A(s_1, x_2)) = f(s_2) = \bar{s}_2$
- $\vdots$
- k. $\bar{r}_k = n_B(\bar{r}_{k-1}, x_k) \underset{k-1. \text{ Zeile}}{=} n_B(\bar{s}_{k-1}, x_k) = \overline{n_A(s_{k-1}, x_k)} = f(n_A(s_{k-1}, x_k)) = f(s_k) = \bar{s}_k$

Nun zeigen wir: $y_1 \dots y_k = q_1 \dots q_k$.

1. $q_1 = o_B(\bar{s}, x_1) = o_A(s, x_1) = y_1$
2. $q_2 = o_B(\bar{r}_1, x_2) = o_B(\bar{s}_1, x_2) = o_A(s_1, x_2) = y_2$
- $\vdots$
- k. $q_k = o_B(\bar{r}_{k-1}, x_k) = o_B(\bar{s}_{k-1}, x_k) = o_A(s_{k-1}, x_k) = y_k \quad \text{q.e.d.}$

Beh.2: Die Zustände von B sind paarweise inäquivalent.

Seien $\bar{s}, \bar{t}$ bel. verschiedene Zustände von B. Es gibt dann Zustände von A $s \in \bar{s}, t \in \bar{t}$, für die nach Beh.1 gilt: $s \dot{\sim} \bar{s}$ und $t \dot{\sim} \bar{t}$.

Angenommen, es gelte $\bar{s} \dot{\sim} \bar{t}$; dann würde auch gelten $s \dot{\sim} t$; dann lägen a und B in der gleichen Äquivalenzklasse und $\bar{s}$ und $\bar{t}$ wären gleich, im Widerspruch zur Voraussetzung. Damit ist Beh. 2 gezeigt.

Beh.3: Für jeden Automaten C in $\bar{A}$ gibt es einen Homomorphismus von C auf B.

Sei $C = (S_C, X, Y, n_C, o_C)$ ein bel. Automat in $\bar{A}$. Weil $C \dot{\sim} A$ und $A \dot{\sim} B$, so $C \dot{\sim} B$.^{+) Nach Satz 2.2.1 gibt es dann für jedes $s \in S_C$ ein äquivalentes $\bar{s} \in S_B$, und zwar genau eins, denn die Zustände von B sind nach Beh.2 paarweise inäquivalent.}

Sei nun f die Abbildung, die jedem $s \in S_C$ den ihm äquivalenten Zustand von B zuordnet. Wir zeigen, daß f ein Homomorphismus von C auf B ist, daß also für bel. $s \in S_C, x \in X$ gilt:

- a) $o_C(s, x) = o_B(f(s), x)$,
- b) $f(n_C(s, x)) = n_B(f(s), x)$ und
- c) alle $t \in S_B$ als f-Bilder vorkommen.

^{+) cf. die Bemerkung nach Def.2.1.5, p.14}

(a) folgt sofort aus der Definition von f . Da für bel. $s \in S_C$: $s \dot{\sim} f(s)$, so folgt nach Lemma 2.2.1 $n_C(s, x) \dot{\sim} n_B(f(s), x)$ für bel. $x \in X$. $f(n_C(s, x))$ ist die Äquivalenzklasse, in der $n_C(s, x)$ nach Def. v. f liegt. Nach Konstruktion von B ist dann $f(n_C(s, x)) = n_B(f(s), x)$; also gilt (b). f ist offenbar eine eindeutige Abbildung von S_C auf S_B , und deshalb gilt auch (c). Damit ist Beh.3 gezeigt.

Beh.4: In $\bar{A}$ gibt es keinen Automaten mit weniger Zuständen als B .

Angenommen, es gibt einen Automaten C in $\bar{A}$ mit weniger Zuständen als B . Dann gibt es nach Beh.3 einen Homomorphismus von C auf B . Dies ist offenbar nicht möglich. Damit ist Beh.4 gezeigt und somit Theorem 2.2.1 vollständig bewiesen.

Da die Zustände von B paarweise inäquivalent sind und so nur zu ebensovielen Äquivalenzklassen zusammengefaßt werden können, wie es Zustände von B gibt, liegt die folgende Definition nahe:

Def.2.2.1 Ein Automat heißt reduziert (minimal), wenn seine Zustände paarweise inäquivalent sind.

2.3 MOORE-Automaten als spezielle MEALY-Automaten

Es erweist sich als praktisch, MEALY-Automaten gewissen Einschränkungen zu unterwerfen. Man gelangt so zu dem äquivalenten Konzept der MOORE-Automaten, das sich theoretisch leichter handhaben läßt. Es sei schon hier darauf hingewiesen, daß die Grammatiken, die wir später benutzen, cum grano salis als spezielle MOORE-Automaten angesehen werden können.

Def.2.3.1 Ein MEALY-Automat $A=(S, X, Y, n, o)$ heißt MOORE-Automat, wenn sich seine Output-Funktion o in der Form

$$o(s, x) = m(n(s, x))$$

darstellen läßt für bel. $s \in S$, $x \in X$. m ist eine eindeutige Abbildung aus S in Y . m heißt Markierungsfunktion von A , $m(s)$ die Markierung des Zustands s .

Der Wert der Output-Funktion o hängt hier also nicht mehr von einem Zustand und einem Eingabsymbol ab, sondern nur noch von einem Zustand. Dennoch leisten MOORE-Automaten dasselbe wie MEALY-Automaten. Es gilt nämlich:

Satz 2.3.1 Zu jedem MEALY-Automat gibt es einen äquivalenten MOORE-Automaten.

Der Satz ist bei GLUSCHKOW, p.18 ff. bewiesen. Für einen endlichen MEALY-Automaten A mit m Zuständen und einem n -elementigen Eingabealphabet kann man einen äquivalenten MOORE-Automaten B effektiv berechnen. B hat $(m+1)n$ Zustände. Bei der Konstruktion von B sorgt man dafür, daß jeder Zustand von B beim Lesen höchstens eines Eingabesymbols erreicht werden kann. Stellt man sich B als Graphen vor, so führen in einen Zustand von B nur Pfeile, die mit demselben Eingabesymbol beschriftet sind. Man kann dann den neuen Zustand als Gedächtnis dafür benutzen, welches Symbol B beim Übergang von dem alten in den neuen Zustand gelesen hat.

Für MOORE-Automaten gelten analoge Äquivalenzsätze wie für MEALY-Automaten. Dennoch empfiehlt es sich, einen stärkeren Äquivalenzbegriff als für MEALY-Automaten einzuführen, denn die Eigenschaft, MOORE-Automat zu sein, geht bei homomorphen Abbildungen von Automaten im allgemeinen verloren. Wir werden daher den Begriff der MOORE-Äquivalenz einführen. Zuvor sei noch gesagt, daß die Markierungsfunktion m eines MOORE-Automaten nur für die Zustände definiert sein muß, die im Wertebereich der Überföhrungsfunktion n liegen. Im folgenden wollen wir nur MOORE-Automaten betrachten, deren Markierungsfunktion für alle Zustände definiert ist. Dies bedeutet keine Einschränkung, denn man kann für unmarkierte Zustände die Markierung in beliebiger Weise festlegen, ohne dadurch das Verhalten eines Automaten zu beeinflussen, denn unmarkierte Zustände spielen für die Output-Funktion eines Automaten keine Rolle. Ein MOORE-Automat ist außerdem durch Angabe seiner Überföhrungsfunktion und seiner Markierungsfunktion sowie seiner Alphabete bereits zureichend definiert. Die Output-Funktion o läßt sich nämlich durch Einsetzung von n in m gewinnen; es gilt: $o(s,x)=m(n(s,x))$, wenn m die Markierungs-, n die Überföhrungsfunktion, s ein bel. Zustand und x ein bel. Symbol des Eingabealphabetes eines MOORE-Automaten sind.

Def. 2.3.2 Ein MOORE-Homomorphismus des MOORE-Automaten

$A = (S_A, X, Y, n_A, m_A)$ auf den MOORE-Automaten

$B = (S_B, X, Y, n_B, m_B)$ ist eine eindeutige Abbildung

f von S_A auf S_B , für die gilt:

1) $f(n_A(s, x)) = n_B(f(s), x)$ für bel. $s \in S_A, x \in X$.

2) $m_A(s) = m_B(f(s))$ für alle $s \in S_A$.

3) $m(s_0^A) = s_0^B$, falls A und B initiale Automaten mit den Anfangszuständen s_0^A bzw. s_0^B sind.

(Auch im folgenden wollen wir Anfangszustände in dieser Weise bezeichnen.)

Def. 2.3.3 Ein Zustand s von A ist MOORE-äquivalent (MO-äq)

dem Zustand t von B g.d.w.

1) $s \text{ äq } t$ und

2) $m_A(s) = m_B(t)$.

Zwei Automaten A und B heißen MOORE-äquivalent

g.d.w. es für jeden Zustand von A einen MOORE-

äquivalenten Zustand von B und für jeden Zustand

von B einen MOORE-äquivalenten Zustand von A

gibt.

Als Folgerungen aus den in 2.2 gebrachten Sätzen haben wir:

Korollar 2.3.1 MOORE-homomorphe Automaten sind MOORE-äquivalent.

Es handelt sich um eine direkte Folgerung aus Satz 2.2.2.

Außerdem gilt als

Korollar 2.3.2 das Reduktionstheorem für MOORE-Automaten,

das genau wie Theorem 2.2.1 formuliert ist, nur das an-

stelle von MEALY-Automaten von MOORE-Automaten, anstelle

von Äquivalenz von MOORE-Äquivalenz⁺⁺⁾ die Rede ist.

Der Beweis wird ganz analog zu dem Beweis von Theorem

2.2.1 geführt. Analog zu Def. 2.2.1 haben wir:

Def. 2.2.4 Ein MOORE-Automat heißt MOORE-reduziert (MOORE-

minimal), wenn seine Zustände paarweise MOORE-

inäquivalent sind.

Endliche MOORE-Automaten können wieder durch Matrizen ^{+))}

dargestellt werden, wobei in der ersten Spalte der jeweiligen

Matrix die Markierungen für die Zustände stehen und die

übrigen Felder die Werte der Überföhrungsfunktion enthalten.

Wir bringen als Beispiel zwei MOORE-Automaten mit s_1, s_2 als

Zuständen, mit x, y als Eingabe- und u, v als Ausgabesymbolen.

^{+))} cf. p.11 f.

⁺⁺⁾ 'und anstelle von Homomorphie von MOORE-Homomorphie' einfügen!

		x	y
s ₁	u	s ₂	s ₂
s ₂	v	s ₂	s ₂

		x	y
s ₁	v	s ₂	s ₂
s ₂	v	s ₂	s ₂

Man erkennt leicht, daß diese beiden Automaten zwar isomorph, aber nicht MOORE-isomorph, zwar äquivalent, aber nicht MOORE-äquivalent sind. Damit haben wir zugleich ein Beispiel dafür, daß homomorphe Automaten nicht MOORE-homomorph sein müssen, worauf wir zu Anfang dieses Abschnitts hingewiesen haben.^{+))}

Als Graphen kann man diese beiden Automaten folgendermaßen darstellen. Wir beginnen mit dem ersten Automaten:



Neben den Namen für die Zustände ist ihre Markierung angegeben. Sind zwei Zustände durch mehrere Pfeile verbunden, so können wir diese zu einem Pfeil 'zusammenklappen' und ihn mit den entsprechenden Eingabesymbolen versehen. Der zweite Automat ließe sich dann folgendermaßen darstellen:



3. BESCHREIBUNG VON SPRACHEN

3.1 Zur Problemstellung

Im folgenden wollen wir den Übergang von den TRn zu den Akzeptoren vorbereiten. Mit den letzteren werden wir uns genauer beschäftigen, da wir mit ihrer Hilfe Sprachen beschreiben werden. Wir vergegenwärtigen uns dazu noch einmal, daß mit Sprache(über V) jede beliebige Teilmenge von $F(V)$ gemeint sein kann.⁺⁺⁾ In der Literatur zur Automatentheorie werden Sprachen in diesem Sinne meist Ereignisse über V genannt.

Wir sagen von einer Sprache, daß sie beschrieben ist, wenn sie durch eine generative Grammatik oder eine Analysegrammatik gegeben ist.⁺⁺⁺⁾ Um zu zeigen, daß man mit

^{+))} cf. p.19

⁺⁺⁾ cf. p.3

⁺⁺⁺⁾ cf. Def.1.2.6 u. Def.1.2.7, p.4

Automaten Sprachen beschreiben kann, müssen wir also zeigen, wie man Automaten als Grammatiken ansehen kann. Dies ist bisher noch nicht einsichtig, denn wir hatten uns klar gemacht, daß Automaten lediglich Verfahren sind, die Ketten in Ketten übersetzen.⁺) Wir deuten nun an, wie man sich einen Automaten als Grammatik vorstellen kann.

Dazu schreibt man auf das Eingabeband des Automaten beliebige Ketten über dem Vokabular V der zu beschreibenden Sprache L_V . Das Eingabealphabet des Automaten sei ebenfalls V . Der Automat übersetzt dann die Eingabeketten in Ketten über seinem Ausgabealphabet. Nun wollen wir genau diejenigen Ketten als zu der durch den Automaten beschriebenen Sprache ansehen, die von dem Automaten in eine Kette übersetzt werden, die auf einen festen Buchstaben endet. Wenn die so durch den Automaten festgelegte Sprache gleich der zu beschreibenden Sprache L_V ist, so ist anschaulich klar, daß wir hier ein Entscheidungsverfahren für L_V vorliegen haben, falls der Automat finit gegeben ist. Die Ausgabeketten sind nämlich endlicher Länge, und deshalb läßt sich für eine beliebige Ausgabekette entscheiden, ob sie auf ein festes Symbol endet oder nicht.⁺⁺⁾

Es sei hier darauf hingewiesen, daß man Sprachen analog zu der zuletzt besprochenen Weise charakterisieren kann, ohne daß sie aufzählbar oder gar entscheidbar wären; man kann in diesem Sinne sogar alle Sprachen charakterisieren⁺⁺⁺⁾, wenn man auf die finite Gegebenheit von Automaten verzichtet. Man postuliert so z.B. die Existenz von Automaten mit z.B. überabzählbar unendlich vielen Zuständen, die aber natürlich nicht i.a. finit gegeben werden können.⁺⁺⁺⁾ Mit derartigen Systemen werden wir es im folgenden zu tun haben. Weshalb wir Sprachen nicht sofort durch Akzeptoren charakterisieren, was intuitiv näherliegend wäre, werden wir in 3.2 motivieren.

⁺) cf. p.11 f.

⁺⁺⁾ Die hier angedeutete Konzeption von Beschreibung wird durch Def.3.2.1 präzisiert.

⁺⁺⁺⁾ Die finite Gegebenheit war eine der Grundforderungen, die wir an eine Grammatik gestellt hatten; cf. p.4. Ein infinites Verfahren ist insbesondere nicht durch einen endlichen Text gegeben, bzw. durch eine endliche Vorschrift, wie der das Verfahren beschreibende Text hergestellt werden kann.

⁺⁺⁺⁺⁾ cf. 3.2

Der im folgenden benutzte Begriff der Charakterisierbarkeit (später technisch 'Darstellbarkeit' genannt, cf. Def.3.2.1) ist allerdings sehr allgemein, da sich, wie schon gesagt, alle Sprachen in diesem Sinne charakterisieren bzw. darstellen lassen. Wir wollen uns nun aber noch kurz überlegen, daß nicht jede Sprache über einem festen Vokabular V durch eine generative Grammatik über V oder gar eine Analysegrammatik über V gegeben werden kann. Eine generative Grammatik über V ist nämlich nach Def.1.2.6 ein finites Verfahren. Das besagt insbesondere, daß G durch einen endlichen Text gegeben sein muß, den man als endliche Zeichenfolge auffassen kann. daraus kann man folgern, daß die Menge aller möglichen generativen Grammatiken über V abzählbar unendlich ist. Die Menge aller Sprachen über V hat aber die Mächtigkeit von $P(V)$, d.h. sie ist überabzählbar unendlich. Es gibt also mehr Sprachen über V als generative Grammatiken über V . Nun definiert eine generative Grammatik aber eindeutig eine Sprache, d.h. durch generative Grammatiken über V lassen sich höchstens abzählbar unendlich viele Sprachen über V beschreiben. Es muß also Sprachen über V geben, die sich nicht durch generative Grammatiken über V beschreiben lassen (cf. CHOMSKY 1958, p.84). Diesen Sachverhalt muß man sich im folgenden ständig vor Augen halten. Wenn wir im nächsten Abschnitt also von 'darstellbaren' bzw. 'charakterisierbaren' Sprachen reden, können damit unter Umständen pathologische Objekte gemeint sein, die nicht effektiv konstruiert werden können.

3.2 Die Darstellbarkeit aller Sprachen durch Automaten

Def.3.2.1 Eine Sprache L über V wird in dem initialen Automaten $A=(S,V,Y,n,o)$, dessen Anfangszustand s_0 ist, durch das Ausgabesymbol $y \in Y$ dargestellt, wenn für alle $p \in F(V)$ gilt:
 $p \in L$ g.d.w. der letzte Buchstabe von $O(s_0,p)$ gleich y ist.

Unser nächstes Ziel ist das

Theorem 3.2.1 Jede nichtleere Sprache über V läßt sich durch ein Ausgabesymbol eines initialen MOORE-Automaten darstellen.

Da es nach dem im vorigen Abschnitt Gesagten Sprachen gibt, die sich nicht durch generative Grammatiken beschreiben lassen, ist der hier benutzte Darstellungsbegriff allgemeiner als der oben eingeführte Beschreibungsbegriff.⁺) Ein MOORE-Automat ist also i.a. kein finites Verfahren, und dies wird unseren Übergang zu den endlichen Automaten und Akzeptoren motivieren.

Da Theorem 3.2.1 nicht zentral zu unserem Thema gehört, verzichten wir auf einen expliziten Beweis und geben lediglich einen Überblick über den Beweisgang. Den Beweis findet man bei GLUSCHKOW, § 2.

Def.3.2.2 Seien X, Y nichtleere Mengen. Eine eindeutige Abbildung f von $F(X)$ in $F(Y)$ heißt darstellbar durch den Automaten $A=(S, X, Y, n, o)$, wenn es einen Zustand $s \in S$ gibt, so daß für alle $p \in F(X)$ gilt:
$$O(s, p) = f(p).$$

Wir sagen: f wird in A durch s dargestellt.

(Offenbar gilt: $O(s, p) = f(p)$ g.d.w. $A_s(p) = \{f(p)\}$, d.h. f muß gleich der durch s von A festgelegten Transducerabbildung sein. ++))

Jede durch A darstellbare Abbildung von $F(X)$ in $F(Y)$ muß die folgenden beiden Bedingungen - Automatenbedingungen genannt - erfüllen.

Automatenbedingung 1: $l(p) = l(f(p))$ für alle $p \in F(X)$.

Automatenbedingung 2: Für bel. $p, q \in F(X)$ ist $f(pq) = f(p)r$, wobei $r = g(pq)$ ein durch das Wort pq eindeutig bestimmtes Wort von $F(Y)$ ist.

Def.3.2.3 Eine eindeutige Abbildung von $F(X)$ in $F(Y)$ heißt Automatenabbildung, wenn sie die Automatenbedingungen erfüllt.

Def.3.2.4 Sei B eine bel., nichtleere Menge, X, Y bel. Vokabulare. Ein MOORE-Automat $A=(S, X, Y, n, m)$, dessen Zustandsmenge S aus allen Wörtern bp mit $b \in B$ und $p \in F(X)$ besteht, dessen Überföhrungsfunktion $n(bp, x) = bpx$ für bel. $bp \in S, x \in X$ ist und dessen Markierungsfunktion m beliebig gewählt sein kann, heißt freier MOORE-Automat.

cf. p.21

++) cf. p.7 und p.14. $A_s = f$ natürlich nur dann, wenn wir wieder wie auf p.14 die Elemente des Bildbereichs von A_s , die sämtlich Einermengen sind, mit den Elementen identifizieren, die sie enthalten.

Lemma 3.2.1 Zu jeder Menge $\{f_a : a \in M\}$ (wobei M eine beliebige, nichtleere Menge ist) von Automatenabbildungen von $F(X)$ in $F(Y)$ gibt es einen freien MOORE-Automaten $A=(S, X, Y, n, m)$, durch den sich jede der Abbildungen f_a darstellen läßt.

Es ist klar, daß die Anzahl der Zustände von A unendlich, eventuell sogar überabzählbar unendlich sein kann.⁺)

Die Automatenbedingungen stellen nun keine echten Einschränkungen für eindeutige Abbildungen von $F(X)$ in $F(Y)$ dar, denn es gilt:

Lemma 3.2.2 Seien X und Y beliebige, nichtleere Mengen. Dann gibt es zu jeder eindeutigen Abbildung f von $F(X)$ in $F(Y)$ eine Automatenabbildung g von $F(X \cup \{a\})$ in $F(Y \cup \{a\})$, aus der sich f in eindeutiger Weise zurückgewinnen läßt ($a \notin X \cup Y$).

(Der Beweis benutzt ein Kodierungsverfahren zur Herstellung der Gleichheit der Wortlängen eines bel. Arguments $p \in F(X)$ und seines f -Bildes. Man fügt dazu rechts an das Wort p m a 's an, wenn $m=l(f(p))$; dann fügt man an das Wort $f(p)$ links n a 's an, wenn $n=l(p)$. Für die so gewonnenen Mengen von gleichlangen Worten⁺⁺) definiert man eine Automatenabbildung g , aus der sich f wieder zurückgewinnen läßt.)

Def.3.2.5 Sei f eine Automatenabbildung von $F(X)$ in $F(Y)$; dann ist S_y die Menge der Ketten über X , deren f -Bild mit dem Symbol y endet.

O.E.d.A. gilt: $S_y \neq \emptyset$, denn falls y in einem f -Bild vorkommt, kommt es wegen Automatenbedingung 1 auch am Ende eines f -Bildes vor. Wenn es an keinem Ende eines f -Bildes vorkommt, so kann man es aus der Menge Y fortlassen.

Ferner sind offenbar alle S_y disjunkt, und die Vereinigung aller S_y ist gleich $F(X) - \{e\}$. Es gilt also:

Lemma 3.2.3 Zu jeder Automatenabbildung von $F(X)$ in $F(Y)$ gibt es ein eindeutig bestimmtes System $M := \{S_y : y \in Y\}$ von Teilmengen von $F(X)$.

Dies legt die folgenden Definitionen nahe:

Def.3.2.6 Sei f eine Automatenabbildung von $F(X)$ in $F(Y)$; dann heißt die Menge M (vgl. Lemma 3.2.3) das von f erzeugte System von Sprachen über X .

⁺) cf. p.22

⁺⁺) d.h. zwei sich jeweils entsprechenden Worte der beiden Mengen sind

Def.3.2.7 Eine Menge von paarweise disjunkten, nichtleeren Sprachen über X , deren Vereinigung gleich $F(X) - \{e\}$ ist, heißt Automatensystem von Sprachen über X .

Es gilt nun:

Lemma 3.2.4 Zu jedem Automatensystem $M = \{S_y : y \in Y\}$ von Sprachen über X gibt es eine Automatenabbildung f von $F(X)$ in $F(Y)$, die M erzeugt.

Nun können wir zeigen:

Lemma 3.2.5 Jedes Automatensystem M von Sprachen über X läßt sich durch einen initialen MOORE-Automaten mit dem Eingabealphabet X darstellen.

Beweis: Nach Lemma 3.2.4 gibt es zunächst eine Automatenabbildung, die M erzeugt. Sei f diese Abbildung. Nach Lemma 3.2.1 kann f durch den Anfangszustand eines freien initialen MOORE-Automaten dargestellt werden (daß man den Anfangszustand dazu wählen kann, ergibt sich aus dem Beweis von Lemma 3.2.1, der hier nicht gebracht worden ist; cf. GLUSCHKOW, p.23, Theorem 7.)

Aus Lemma 3.2.5 folgt sofort Theorem 3.2.1, Denn, um eine Sprache L über V in einem initialen MOORE-Automaten darzustellen, müssen wir nur das Automatensystem $\{L, -L\}$ wählen, das aus L und $-L$, dem Komplement von L bezüglich $F(V)$ besteht. (Die leere Kette soll dabei nicht in L und nicht in $-L$ liegen, dh. strenggenommen müßte man $L - \{e\}$ schreiben, falls $e \in L$. Wir wollen diesen Fall aber als unwesentlich ansehen.) Nach Lemma 3.2.5 läßt sich dies Automatensystem in einem initialen MOORE-Automaten mit dem Eingabealphabet V darstellen. Damit ist L auch durch einen bestimmten Ausgabebuchstaben in diesem Automaten dargestellt.

Der Grund, weshalb man die Darstellbarkeit von Sprachen nicht sofort mit Hilfe von Akzeptoren beweist, liegt darin, daß man durch sehr einfache Eigenschaften der Automatenabbildungen (die in den Automatenbedingungen zum Ausdruck kommen) und der Bildmengen von Automatenabbildungen sehr komplizierte Argumentmengen festlegen kann. Eigenschaften wie "der letzte Buchstabe einer Kette",⁺ mit deren Hilfe z.B. die Mengen S_y definiert sind, sind nämlich elementar ent-

⁺ 'Buchstabe' wird hier und auch an anderer Stelle synonym zu 'Symbol' gebraucht.

scheidbar. Dies nutzt man für die Beweise aus.⁺)

Daß man für die Darstellung von Sprachen aber grundsätzlich auf das Ausgabealphabet verzichten kann, zeigen wir im folgenden Abschnitt, der so einen organischen Übergang von den Automaten zu den Akzeptoren herstellt.

3.3 Die Darstellbarkeit von Sprachen in Automaten durch Zustände

Wir werden im folgenden zeigen, daß es für initiale MOORE-Automaten genügt, Sprachen durch Zustände zu charakterisieren, und nicht, wie bisher, durch Ausgabebuchstaben. Dazu erweist es sich als praktisch, eine weitere verallgemeinerte Überföhrungsfunktion zu definieren.

Def.3.3.1 Sei $A=(S,X,Y,n,o)$ ein bel. Automat;

dann definieren wir für alle $s \in S$, $x \in X$,

$x_1 \dots x_k \in F(X)$:

$M(s,x) := n(s,x)$

$M(s,e) := s$

$M(s,x_1 \dots x_k) := n(M(s,x_1 \dots x_{k-1}), x_k)$

Wir sagen: A wird durch die Kette p aus s in t überföhrt, falls $t=M(s,p)$

(M soll an engl. to move erinnern.).

Man kann sich überlegen, daß für alle i , $1 \leq i \leq k$, für alle Zustände s eines Automaten A und alle

$x_1 \dots x_k \in F(X)$ gilt:

1) $M(s,x_1 \dots x_k) = M(M(s,x_1 \dots x_i), x_{i+1} \dots x_k)$ und

2) $O(s,x_1 \dots x_k) = O(s,x_1 \dots x_i) O(M(s,x_1 \dots x_i), x_{i+1} \dots x_k)$.

Zu 1: Es genügt zu zeigen, daß für bel. Ketten $z_1 \dots z_m \in$

$F(X)$, $s \in S$ gilt:

$M(M(s,x_1 \dots x_k), z_1 \dots z_1) = M(M(s,x_1 \dots x_k z_1), z_2 \dots z_m)$.

Bew. durch Induktion über die Länge von $z_1 \dots z_m$:

Den I.A. erhält man aus der Definition von M .

I.S.: $M(M(s,x_1 \dots x_k), z_1 \dots z_m)$

$= n(M(M(s,x_1 \dots x_k), z_1 \dots z_{m-1}), z_m)$ nach Def. von M

$= n(M(M(s,x_1 \dots x_k z_1), z_2 \dots z_{m-1}), z_m)$ nach I.V.

$= M(M(s,x_1 \dots x_k z_1), z_2 \dots z_m)$ nach Def. von M .

Zu 2: Die Behauptung folgt sofort aus der Def. von O und M .

⁺) cf. die Bemerkung auf p.22 unten.

Im folgenden werden wir stets initiale Automaten bzw. Akzeptoren mit dem Anfangszustand s_0 betrachten.

Die folgende Definition ist praktisch ein Spezialfall von Def. 1.3.7; sie erlaubt es, einen Automaten ohne Rücksicht auf sein Ausgabealphabet als Akzeptor anzusehen.

Def.3.3.2 Die Sprache L über X wird in dem initialen Automaten A durch die Menge F von Zuständen festgelegt, wenn X das Eingabealphabet von A ist und L genau aus den Wörtern (Sätzen) besteht, die A aus dem Anfangszustand s_0 in einen Zustand aus F überführen. Wir nennen die Zustände in F auch Endzustände von A .

Läßt man das Ausgabealphabet und die Outputfunktion von A weg, so erhalten wir einen Akzeptor A' . L wird in A durch F offenbar genau dann festgelegt, wenn $L = L_{\{s_0\}}^F$, wobei sich die letzte Notation auf den Akzeptor A' bezieht.⁺⁾

Nun können wir den folgenden Satz formulieren:

Satz 3.3.1 1) Vor.: Sei L eine bel. Sprache, die in dem initialen MOORE-Automaten A durch das Ausgabesymbol y dargestellt wird.⁺⁺⁾

Beh.: Die Menge F der durch y markierten Zustände von A stellt $L \cup \{e\}$ bzw. L dar, je nachdem, ob der Anfangszustand von A durch y markiert ist oder nicht.

2) Vor.: Sei L eine bel. Sprache, die in dem initialen MOORE-Automaten A durch eine nicht-leere Menge F von Zuständen dargestellt wird.

Beh.: $L - \{e\}$ bzw. L wird in dem initialen MOORE-Automaten A' , den man erhält, indem man die Markierungsfunktion von A so abändert, daß sie den Zuständen von F und nur diesen, ein festes y zuordnet, durch das Ausgabesymbol y dargestellt, je nachdem, ob $e \in L$ oder e nicht $\in L$.

Zu 1: Sei L eine Sprache über X , die in dem initialen MOORE-Automaten A durch y festgelegt wird. Sei F die Menge der durch y markierten Zustände von A . Sei nun $x_1 \dots x_k$ ein bel. Eingabewort von A , $s_1 \dots s_k = N(s_0, x_1 \dots x_k)$ und $y_1 \dots y_k = O(s_0, x_1 \dots x_k)$. Man überzeugt sich leicht, daß $m(s_k) = y_k$ ist. $x_1 \dots x_k$ gehört aber nur dann zu L , wenn $y_k = y$. Dies ist genau dann der Fall,

⁺⁾ cf. Def.1.3.7, p.9 ⁺⁺⁾ cf. Def.3.2.1, p.23

wenn $x_1 \dots x_k$ A aus s_0 in einen mit y markierten Zustand - also einen Zustand in F - überführt. Durch F wird also eine Sprache L' dargestellt, die alle Wörter von L enthält. L' kann darüber hinaus noch das leere Wort e enthalten, und zwar genau dann, wenn s_0 durch y markiert ist und somit in F liegt.⁺⁾

Zu 2: Sei L' eine Sprache über X, die in dem initialen MOORE-Automaten $A=(S,X,Y,n,m)$ durch die nichtleere Menge $F \subseteq S$ dargestellt wird. Wir konstruieren den initialen MOORE-Automaten $A'=(S,X,\{y,y'\},n,m')$ mit demselben Anfangszustand s_0 wie A, dem Ausgabealphabet $\{y,y'\}$, wobei die beiden Elemente bel. gewählt werden können, aber verschieden sein müssen, und der Markierungsfunktion

$$m'(s) := \begin{cases} y, & \text{falls } s \in F \\ y' & \text{sonst} \end{cases}$$

für bel. $s \in S$. Wir nennen nun die Sprache, die in A' durch das Ausgabesymbol y dargestellt wird, L. Man kann sich leicht davon überzeugen, daß jedes nichtleere Eingewort genau dann zu L gehört, wenn es auch zu L' gehört. Falls s_0 in A zu F gehört, so enthält L' allerdings auch das leere Wort e. L kann aber e nicht enthalten, denn dann müßte $O(s_0,e)=y$ gelten, wobei O die verallgemeinerte Output-Funktion von A' ist. Es gilt aber: $O(s_0,e)=s_0$. Wiederum unterscheiden sich also L' und L höchstens um das leere Wort.

Wir halten als Ergebnis dieses Abschnitts fest, daß genügt, Sprachen durch Zustandsmengen in initialen MOORE-Automaten darzustellen. Daß sich zwei Sprachen eventuell um das leere Wort unterscheiden, wollen wir als unwesentlich ansehen. Als Nebenergebnis haben diese Überlegungen ergeben, daß man mit einem Ausgabealphabet von zwei Buchstaben alle Sprachen in MOORE-Automaten darstellen kann. ⁺⁺⁾

^{+) 'Wort' wird hier und an anderer Stelle synonym zu 'Kette' gebraucht.}

⁺⁺⁾ Wir reden aus stilistischen Gründen oft von der Darstellbarkeit von Sprachen in Automaten, statt durch Automaten.

4. FINITE-STATE-AKZEPTOREN UND -SPRACHEN

Satz 3.3.1 hat gezeigt, daß man Sprachen durch MOORE-Automaten ohne Zurhilfenahme der Output-Funktion beschreiben kann. Deshalb kann man auf das Ausgabealphabet für die Sprachbeschreibung prinzipiell verzichten und gelangt so zu den Akzeptoren.

Da sich unbeschränkte Akzeptoren ⁺) technisch nicht verwirklichen lassen, wollen wir im folgenden Finite-State-Akzeptoren (FSAen) zur Sprachbeschreibung einführen.

Jeder FSA kann mutatis mutandis als spezieller MOORE-Automat angesehen werden, d.h. man kann ihn sofort effektiv in einen MOORE-Automaten umschreiben. ⁺⁺) Dies hat zur Folge, daß man die für MOORE-Automaten bewiesenen Sätze auf die FSAen übertragen kann.

4.1 Determinierte und nichtdeterminierte Finite-State-Akzeptoren

Im folgenden führen wir zwei Arten von FSAen ein, die wir kurz endliche (finite) Akzeptoren (FAen) nennen, nämlich die determinierten und die nichtdeterminierten Finite-State-Akzeptoren.

Def.4.1.0 Ein determinierter Finite-State-Akzeptor (DFA)

$A=(S,V,n,s_0,F)$ ist ein Fünftupel, wobei

1) S eine endliche, nichtleere Menge ist (die Zustandsmenge von A),

2) V ein endliches Vokabular (das V. von A),

3) n eine eindeutige Abbildung von $S \times V$ in S (die Überföhrungsfunktion von A) ist,

4) $s_0 \in S$ ein ausgezeichneter Zustand (der Anfangszustand von A) und

5) $F \subseteq S$ (die Menge der Endzustände von A) ist.

Def.4.1.1 Eine Kette $p \in F(V)$ wird von dem DFA $A=(S,V,n,s_0,F)$ akzeptiert g.d.w. $M(s_0,p) \in F$. ⁺⁺⁺) $L(A)$ ist die Menge der von A akzeptierten Ketten; sie heit die durch A beschriebene Sprache.

⁺) cf. Def.1.3.6 ⁺⁺) cf. p.31

⁺⁺⁺) M ist ebenso wie für Automaten definiert; cf. Def.3.3.1, p.27.

Wir können uns schnell klarmachen, daß man einen DFA unter Beachtung der folgenden Konventionen als MOORE-Automaten auffassen kann. Sei also $A=(S,V,n,s_0,F)$ ein DFA. Wir konstruieren dann einen endlichen MOORE-Automaten $A'=(S,V,\{0,1\},n,m)$ mit der Markierungsfunktion

$$m(s) := \begin{cases} 1, & \text{falls } s \in F \\ 0 & \text{sonst} \end{cases}$$

für bel. $s \in S$. Ansonsten ist A' genau wie A definiert. Es ist so klar, daß jeder DFA A in eindeutiger Weise einen initialen MOORE-Automaten A' festlegt, wobei $L(A)$ die Sprache ist, die in A' durch die Menge F oder das Ausgabesymbol 1 festgelegt bzw. dargestellt ist. ⁺⁾

Def.4.1.2 Ein nichtdeterminierter Finite-State-Akzeptor (NFA) $A=(S,V,n,I,F)$ ist ein Fünftupel, wobei

- 1) V ein endliches Vokabular ist,
- 2) S eine endliche Zustandsmenge ist,
- 3) n eine eindeutige Abbildung von $S \times V$ in $P(S)$ (die Überföhrungsfunktion von A) ist,
- 4) $I \subseteq S$, $I \neq \emptyset$ (die Menge der Anfangszustände von A) und
- 5) $F \subseteq S$ (die Menge der Endzustände von A) ist.

Def.4.1.3 Sei $A=(S,V,n,I,F)$ ein NFA.

Eine beliebige Kette $p=x_1 \dots x_k \in F(V)$ wird von A akzeptiert g.d.w. es eine Folge von Zuständen von A $s_0, \dots, s_k$ (die akzeptierende Zustandsfolge von A für p) gibt mit:

- 1) $s_0 \in I$,
- 2) $s_i \in n(s_{i-1}, x_i)$ für alle i , $1 \leq i \leq k$ und
- 3) $s_k \in F$.

Die Menge der von A akzeptierten Ketten nennen wir wiederum $L(A)$.

Def.4.1.4 Zwei FAen A und B heißen äquivalent g.d.w. $L(A)=L(B)$.

Wir bemerken zu den vorhergehenden Definitionen folgendes: Offensichtlich kann jeder DFA als NFA angesehen werden; der Wertebereich der Überföhrungsfunktion eines DFAs kann nämlich als Spezialfall nur einelementige Mengen enthalten, die wir mit den Elementen, die sie enthalten identifizieren können. Wir wollen ferner festlegen: $e \in L(A)$ g.d.w. $I \cap F \neq \emptyset$.

⁺⁾ Es handelt sich hier wieder um die auf p.29 gebrachte Konstruktion.

Analog zu den Automaten können wir uns FAen wiederum als Mechanismen vorstellen. FAen haben allerdings lediglich ein Eingabeband, daß sich in einer Richtung, z.B. von rechts nach links bewegt. Ein FA beginnt im Anfangszustand zu arbeiten; er liest jeweils einen Buchstaben auf dem Band und geht dann in Abhängigkeit von diesem Buchstaben und dem Zustand, in dem er sich befindet, in einen neuen Zustand über. Die Übergänge werden durch die Überföhrungsfunktion n beschrieben. Es ist klar, daß sich n durch eine n -Matrix beschreiben läßt.⁺) Analog zu dem in 2.1 Gesagten⁺⁺) können FAen natürlich auch durch Graphen beschrieben werden.

Auf den ersten Blick könnte man vermuten, daß ein NFA i.a. ein stärkeres Verfahren als ein DFA ist, denn ein NFA kann von einem Zustand in Abhängigkeit von dem gerade gelesenen Symbol und diesem Zustand eventuell in verschiedene Zustände übergehen. Diese Fähigkeit ist mit dem Wort 'nichtdeterminiert' angesprochen. Dennoch lassen sich durch NFAen nicht mehr Sprachen beschreiben als durch DFAen. Es gilt nämlich Theorem 4.1.1 Zu jedem NFA gibt es einen äquivalenten DFA. Zum Beweis erweist sich die folgende Definition als praktisch:
Def.4.1.5 Sei $A=(S,V,n,I,F)$ ein NFA.

Sei dann $D(A)$ der DFA $(T,V,n_{D(A)},I,F_{D(A)})$ mit

1) $T:=P(S)$,

2) $n_{D(A)}$ ist eine Abbildung von $T \times V$ in T , wobei für bel. $t \in T, x \in V$ gilt:

$$n_{D(A)}(t,x) := \bigcup_{s \in t} n(s,x) \text{ und}$$

4) $F_{D(A)} := \{ y \in T : \text{es gibt ein } z \in y : z \in F \}$.

Beweis von Theorem 4.1.1 ⁺⁺⁺⁾

Sei $A=(S,V,n,I,F)$ ein bel. NFA. Da $D(A)$ offenbar ein DFA ist, genügt es zu zeigen: $L(A)=L(D(A))$.

Beh.1: $L(A) \subseteq L(D(A))$

Bew. v. Beh.1: Sei $p=x_1 \dots x_k \in F(V)$ ein bel. Satz von $L(A)$ mit der akzeptierenden Zustandsfolge $s_0, \dots, s_k$. Wir zeigen

$$s_k \in M_{D(A)}(I, x_1 \dots x_k),$$

wobei I der Anfangszustand von $D(A)$ ist. Damit ist auch

⁺) cf. p.11 ⁺⁺) cf. p.12

⁺⁺⁺) Der Beweis findet sich u.a. bei GINSBURG 1962, p.112 f.

Beh.1 bewiesen, denn s_k ist nach Voraussetzung in F , weshalb nach Konstruktion von $D(A)$ auch $M_{D(A)}(I,p) \in F_{D(A)}$ und folglich $p \in L(D(A))$.

1. Nach Vor. gilt:

$s_1 \in n(s_0, x_1)$, $s_0 \in I$, daraus folgt:

$$n(s_0, x_1) \subseteq \bigcup_{s \in I} n(s, x_1) \stackrel{(\text{Def. } n_{D(A)})}{=} M_{D(A)}(I, x_1),$$

folglich $s_1 \in M_{D(A)}(I, x_1)$.

2. Nach Vor. und dem 1. Beweisschritt gilt:

$s_2 \in n(s_1, x_2)$, $s_1 \in M_{D(A)}(I, x_1)$, daraus folgt:

$$\begin{aligned} n(s_1, x_2) &\subseteq \bigcup_{s \in M_{D(A)}(I, x_1)} n(s, x_2) = n_{D(A)}(M_{D(A)}(I, x_1), x_2) \\ &= M_{D(A)}(I, x_1 x_2), \text{ daraus folgt: } s_2 \in M_{D(A)}(I, x_1 x_2). \end{aligned}$$

⋮

k. Nach Vor. und dem k-1-ten Beweisschritt gilt:

$s_k \in n(s_{k-1}, x_k)$, $s_{k-1} \in M_{D(A)}(I, x_1 \dots x_{k-1})$, daraus folgt:

$$\begin{aligned} n(s_{k-1}, x_k) &\subseteq \bigcup_{s \in M_{D(A)}(I, x_1 \dots x_{k-1})} n(s, x_k) \\ &= n_{D(A)}(M_{D(A)}(I, x_1 \dots x_{k-1}), x_k) = M_{D(A)}(I, x_1 \dots x_k), \text{ woraus} \\ &\text{schließlich folgt: } s_k \in M_{D(A)}(I, x_1 \dots x_k) \text{ q.e.d.} \end{aligned}$$

Beh.2: $L(D(A)) \subseteq L(A)$

Sei $p = x_1 \dots x_k$ ein bel. Satz von $L(D(A))$. Wir zeigen, daß es eine akzeptierende Zustandsfolge $s_0, \dots, s_k$ für p in A gibt. Wir führen zunächst die folgende Abkürzung ein: sei für ein bel. Wort $x_1 \dots x_n \in F(V)$: $t_n := M_{D(A)}(I, x_1 \dots x_n)$.

Bew. v. Beh.2: Nach Vor. gibt es zunächst ein $t_k \in F_{D(A)}$,

so daß gilt: $M_{D(A)}(I, p) = t_k$. Sei nun s_k ein bel. Zustand aus $t_k \cap F$. Die Konstruktion von $D(A)$ garantiert uns die Existenz eines solchen s_k . Dann gilt:

$$\begin{aligned} 1. \quad s_k &\in t_k = M_{D(A)}(I, x_1 \dots x_k) \stackrel{(\text{Def. v. M})}{=} n_{D(A)}(t_{k-1}, x_k) \\ &\stackrel{(\text{Def. v. } n_{D(A)})}{=} \bigcup_{s \in t_{k-1}} n(s, x_k), \text{ daraus folgt:} \end{aligned}$$

es gibt ein $s_{k-1} \in t_{k-1}$: $s_k \in n(s_{k-1}, x_k)$.

$$2. s_{k-1} \in t_{k-1} = M_{D(A)}(I, x_1 \dots x_{k-1}) = n_{D(A)}(t_{k-2}, x_{k-1})$$

$$= \bigcup_{s \in t_{k-2}} n(s, x_{k-1}), \text{ daraus folgt:}$$

$$\text{es gibt ein } s_{k-2} \in t_{k-2}: s_{k-1} \in n(s_{k-2}, x_{k-1}).$$

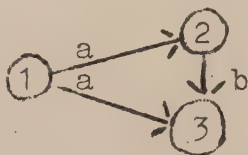
⋮

$$k. s_1 \in t_1 = M_{D(A)}(I, x_1) = \bigcup_{s \in I} n(s, x_1), \text{ woraus folgt:}$$

$$\text{es gibt ein } s_0 \in I: s_1 \in n(s_0, x_1).$$

Wir haben also gezeigt, daß es eine Zustandsfolge $s_k, s_{k-1}, \dots, s_0$ in A gibt, so daß für alle $i, 0 \leq i \leq k$ gilt: $s_i \in n(s_{i-1}, x_i)$, $s_0 \in I$ und $s_k \in F$. Dies ist eine akzeptierende Zustandsfolge für p in A (die Folge muß dabei natürlich von rechts nach links gelesen werden). q.e.d.

Wir veranschaulichen uns die durch Theorem 4.1.1 gegebene Konstruktion an Hand eines Beispiels. Sei $A = (\{1, 2, 3\}, \{a, b\}, n, \{1\}, \{3\})$. Die Definition von n können wir an dem folgenden Graphen ablesen:

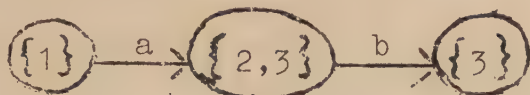


Dann ist $D(A) = (P(\{1, 2, 3\}), \{a, b\}, n_{D(A)}, \{1\}, \{\{1, 3\}, \{2, 3\}, \{3\}, \{1, 2, 3\}\})$. $n_{D(A)}$ ist durch die folgende Matrix gegeben:

	a	b
$\{1, 2, 3\}$	$\{2, 3\}$	$\{3\}$
$\{1, 2\}$	$\{2, 3\}$	$\{3\}$
$\{1, 3\}$	$\{3\}$	$\emptyset$
$\{2, 3\}$	$\emptyset$	$\{3\}$
$\{1\}$	$\{2, 3\}$	$\emptyset$
$\{2\}$	$\{3\}$	$\emptyset$
$\{3\}$	$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$	$\emptyset$

Da der Anfangszustand von $D(A)$ gleich $\{1\}$ ist und nur solche Zustände für die Arbeitsweise von $D(A)$ wichtig sind, die von $\{1\}$ aus erreicht werden können, kann man sich leicht überlegen, daß nur sehr wenige Zustände von $P(\{1, 2, 3\})$ für die Konstruktion von $D(A)$ effektiv benötigt werden.

von $\{1\}$ aus erreichbaren Zustände von $D(A)$ beschreibende Graph sieht folgendermaßen aus:



Man beachte: $\{2,3\}, \{3\} \in F_{D(A)}$. Deshalb sind $a, ab \in L(D(A))$.

Anschaulich gesprochen hat man bei dieser Konstruktion die beiden mit 'a' beschrifteten Pfeile 'zusammengeklappt'.

4.2 Sätze über determinierte Finite-State-Akzeptoren

In 4.1 haben wir gesagt, daß sich jeder DFA als spezieller initialer MOORE-Automat auffassen läßt und sich daher die Sätze für MOORE-Automaten auf DFAen übertragen lassen.⁺)

Es ist jedoch praktisch, die in Frage kommenden Sätze etwas anders zu formulieren, d.h. sie auf DFAen zuzuschneiden.⁺⁺⁾

Def.4.2.1 Seien $A=(S_A, V, n_A, s_0^A, F_A)$, $B=(S_B, V, n_B, s_0^B, F_B)$ bel. DFAen.

Unter einem Homomorphismus von A auf B verstehen

wir eine eindeutige Abbildung f von S_A auf S_B ,

wobei für bel. $x \in V, s \in S_A$ gilt:

- 1) $f(n_A(s, x)) := n_B(f(s), x)$,
- 2) $s \in F_A$ g.d.w. $f(s) \in F_B$ und
- 3) $f(s_0^A) = s_0^B$.

Man macht sich leicht klar, daß Def.4.2.1 eine äquivalente Formulierung für die Definition des MOORE-Homomorphismus f von A' auf B' ist, wobei A' und B' die durch A und B eindeutig festgelegten MOORE-Automaten sind.⁺⁺⁺⁾ Denn für einen MOORE-Homomorphismus muß neben (1) und (3) noch gelten: $m_{A'}(s) = m_{B'}(f(s))$.⁺⁺⁺⁺⁾ Dies ist aber gerade durch (2) garantiert, wie man sich durch einen Blick auf die Definition von m für A' und B' in 4.1 überzeugt.⁺⁺⁺⁺⁺⁾

⁺) Es handelt sich um die in 2.2 und 2.3 gebrachten Sätze.

⁺⁺⁾ Gemeint sind besonders die in 2.3 gebrachten Begriffsbildungen.

⁺⁺⁺⁾ cf. p.31

⁺⁺⁺⁺⁾ cf. Def.2.3.2, p.20. Gemeint ist ein MOORE-Homomorphismus von A' auf B' .

⁺⁺⁺⁺⁺⁾ cf. p.31

Def.4.2.2 Seien $A=(S_A, V, n_A, s_O^A, F_A)$, $B=(S_B, V, n_B, s_O^B, F_B)$

bel. DFAen.

Ein Zustand s von A ist einem Zustand t von B MOORE-äquivalent (s MO-äq t) g.d.w. für bel.

$p \in F(V)$ gilt:

$M_A(s, p) \in F_A$ g.d.w. $M_B(t, p) \in F_B$ (1) und

$s \in F_A$ g.d.w. $t \in F_B$ (2).

Dies ist wiederum eine äquivalente Übertragung der MOORE-Äquivalenz der von A und B festgelegten Automaten

A' und B' ^{+) auf A bzw. B . Zwei Zustände s von A' und t von B' sind nämlich genau dann MOORE-äquivalent, wenn s äq t und $m_{A'}(s) = m_{B'}(t)$. ⁺⁺⁾ Letzteres ist nach dem zu Def.4.2.1 Bemerkten hier durch (2) garantiert.}

Es ist nämlich s äq t g.d.w. $o_{A'}(s, p) = o_{B'}(t, p)$ für bel. $p \in F(V)$. ⁺⁺⁺⁾ Sei nun $p = x_1 \dots x_k$ eine bel. Kette aus $F(V)$.

Nach Def. von o für MOORE-Automaten ⁺⁺⁺⁺⁾ gilt offensichtlich:

$o_I(q, p) = m_I(M_I(q, x_1)) m_I(M_I(q, x_1 x_2)) \dots m_I(M_I(q, x_1 \dots x_k))$
($I=A', B'$, $q=s$, falls $I=A'$, $q=t$, falls $I=B'$, auch i.f.)

Nun ist für bel. $r \in F(V)$: $M_I(q, r) = M_J(q, r)$ ($J= A$, f. $I=A'$, B sonst)

$M_A(s, r) \in F_A$ g.d.w. $M_B(t, r) \in F_B$. Daraus und aus den

Def. der m_I ergibt sich sofort, daß für bel. $r \in F(V)$:

$m_I(M_I(q, r)) = m_I(M_I(q, r))$, und folglich gilt:

$o_{A'}(s, p) = o_{B'}(t, p)$, was wir zeigen wollten.

Def.4.2.3 Zwei FSAen A und B heißen MOORE-äquivalent

(A MO-äq B) g.d.w. es zu jedem Zustand von

A einen MOORE-äquivalenten Zustand von B und

zu jedem Zustand von B einen MOORE-äquivalenten

Zustand von A gibt. Darüberhinaus müssen die

Anfangszustände von A und B MOORE-äquivalent sein.

^{+) cf. p.29}

^{++) cf. Def.2.3.3, p.20}

⁺⁺⁺⁾ cf. p.14

⁺⁺⁺⁺⁾ cf. Def.2.3.1, p.18

Die MOORE-Äquivalenz von DFAen ist eine stärkere Forderung als der für FA i.a. eingeführte Äquivalenzbegriff ⁺). Den Zusammenhang zwischen diesen beiden Äquivalenzbegriffen illustriert der folgende Satz.

Satz 4.2.1 Seien A und B bel. DFAen mit MOORE-äquivalenten Anfangszuständen. Genau in diesem Fall ist $L(A)=L(B)$, und folglich sind A und B äquivalent.

Beweis: Der Satz folgt sofort aus Def.4.2.2. Seien nämlich s_0^A, s_0^B die Anfangszustände von A bzw. B. $s_0^A \text{ MO-} \dot{a}q s_0^B$ g.d.w. für bel. p über dem Vokabular von A oder von B: $M_A(s_0^A, p) \in F_A$ g.d.w. $M_B(s_0^B, p) \in F_B$ und $s_0^A \dot{a}q s_0^B$. Mithin $L(A)=L(B)$, da auch $e \in L(A)$ g.d.w. $e \in L(B)$, weil $s_0^A \in F_A$ g.d.w. $s_0^B \in F_B$.

Aus früher bewiesenen Sätzen ergeben sich die folgenden beiden Korollare:

Korollar 4.2.1 MOORE-homomorphe DFAen sind MOORE-äquivalent. (Direkte Folgerung aus Korollar 2.3.1, p.20)

Korollar 4.2.2 das Reduktionstheorem für DFAen, das genau wie Theorem 2.2.1 formuliert ist, nur daß anstelle von MEALY-Automaten von DFAen, anstelle von Äquivalenz von MOORE-Äquivalenz und anstelle von Homomorphie von MOORE-Homomorphie die Rede ist. ⁺⁺ (Es handelt sich um eine direkte Folgerung aus Korollar 2.3.1, p.20)

Analog zu Def.2.3.4 ⁺⁺⁺) setzen wir wiederum fest:

Def.4.2.4 Ein DFA heißt (MOORE-) reduziert (minimal) g.d.w. seine Zustände paarweise MOORE-inäquivalent sind.

Unser nächstes Ziel ist es zu zeigen, daß man zu jedem DFA A einen reduzierten DFA B effektiv konstruieren kann. Dazu benötigen wir einige Definitionen und Lemmata.

⁺) cf. Def.4.1.4, p.31

⁺⁺) cf. p.15

⁺⁺⁺) cf. p.20

Def.4.2.5 Ein Zustand t eines Automaten oder DFAs A ist vom Zustand s von A aus erreichbar, wenn ein p über dem (Eingabe-) Alphabet von A gibt mit $M_A(s,p)=t$. Ein initialer Automat oder DFA heißt zusammenhängend, wenn jeder Zustand von A von seinem Anfangszustand aus erreichbar ist.

Def.4.2.6 Sei $A=(S,X,Y,n,o)$ ein Automat; dann gilt für für bel. $s,t \in S$:

$s \stackrel{k}{\sim} t$ (s ist k -äquivalent zu t) g.d.w.
 $O(s,p)=O(t,p)$ für alle $p \in F(X)$ der Länge $\leq k$.
 $\stackrel{k}{\sim}$ ist offensichtlich eine Äquivalenzrelation.
 Sei P_k die Menge der durch $\stackrel{k}{\sim}$ erzeugten Äquivalenzklassen; wir nennen P_k auch die Menge der k -Klassen von A .

Sei für das folgende $A=(S,X,Y,n,o)$ ein Automat.

Lemma 4.2.1 $s \stackrel{k+1}{\sim} t$ g.d.w. $s \stackrel{k}{\sim} t$ und $n(s,x) \stackrel{k}{\sim} n(t,x)$ für bel. $x \in X$.

Lemma 4.2.2 Wenn $P_{k+1}=P_k$, so $P_k=P_j$ für alle $j \geq k$.

Beweis von Lemma 4.2.1:

Wir zeigen die Implikation zuerst von links nach rechts.

Vor.: $s \stackrel{k+1}{\sim} t$

Beh.: a) $s \stackrel{k}{\sim} t$

b) $n(s,x) \stackrel{k}{\sim} n(t,x)$ für alle $x \in X$.

Beh. (a) ist trivial. Wir zeigen (b).

Sei x ein bel. Eingabesymbol. Da $s \stackrel{k+1}{\sim} t$, so

$O(s, xp)=O(t, xp)$ für bel. $p \in F(X)$ der Länge k , also

$o(s,x)O(n(s,x),p)=o(t,x)O(n(t,x),p)$. Weil aber $s \stackrel{k+1}{\sim} t$,

so auch $o(s,x)=o(t,x)$, also $O(n(s,x),p)=O(n(t,x),p)$ und folglich $n(s,x) \stackrel{k}{\sim} n(t,x)$. q.e.d.

Nun zeigen wir die umgekehrte Richtung.

Vor.: $s \stackrel{k}{\sim} t$, $n(s,x) \stackrel{k}{\sim} n(t,x)$ für bel. $x \in X$

Beh.: $s \stackrel{k+1}{\sim} t$.

Sei $p=x_1 \dots x_{k+1}$ ein bel. Eingabewort von A ; nach Vor. gilt:

$n(s,x_1) \stackrel{k}{\sim} n(t,x_1)$, also

$O(n(s,x_1), x_2 \dots x_{k+1})=O(n(t,x_1), x_2 \dots x_{k+1})$. Aus $s \stackrel{k}{\sim} t$ folgt:

$o(s,x_1)=o(t,x_1)$, also

$O(s, x_1 \dots x_{k+1})=o(s, x_1)O(n(s, x_1), x_2 \dots x_{k+1})$
 $=o(t, x_1)O(n(t, x_1), x_2 \dots x_{k+1})$
 $=O(t, x_1 \dots x_{k+1}),$

also $s \stackrel{k+1}{\sim} t$. q.e.d.

Beweis von 4.2.2 durch Induktion: Sei $P_k = P_{k+1}$. Als Induktionsvoraussetzung nehmen wir: $P_{k+i} = P_k$ für alle $i, 1 \leq i \leq m, m \geq 1$. Wir zeigen, daß $P_{k+m+1} = P_k$. Dazu genügt es zu zeigen, daß $s \stackrel{k+m}{\sim} t$ g.d.w.

$s \stackrel{k+m+1}{\sim} t$ für bel. Zustände s, t . Seien also s, t bel. Zustände mit $s \stackrel{k+m}{\sim} t$. Dies ist nach Lemma 4.2.1 genau dann der Fall, wenn $s \stackrel{k+m-1}{\sim} t$ und $n(s, x) \stackrel{k+m-1}{\sim} n(t, x)$ für bel. $x \in X$. Weil nach I.V. $P_{k+m-1} = P_{k+m}$, so auch $s \stackrel{k+m}{\sim} t$ und $n(s, x) \stackrel{k+m}{\sim} n(t, x)$. Genau dann ist nach Lemma 4.2.1 $s \stackrel{k+m+1}{\sim} t$: q.e.d.

Nun können wir das folgende Theorem beweisen:

Theorem 4.2.1 Sei A ein endlicher, zusammenhängender MOORE-Automat; dann läßt sich ein zu A MOORE-äquivalenter MOORE-reduzierter Automat effektiv konstruieren.

Wir geben im folgenden einen Algorithmus zur Reduktion (Minimierung) von endlichen MOORE-Automaten an. Damit haben wir zugleich einen Algorithmus zur Reduktion von DFAen. Um einen DFA zu reduzieren, müssen wir nämlich nur den durch ihn eindeutig festgelegten endlichen MOORE-Automaten reduzieren. ^{+) Wir können uns bei der Reduktion auf zusammenhängende Automaten ⁺⁺⁾ beschränken, da wir den Algorithmus speziell zur Reduktion von DFAen anwenden wollen und nicht-erreichbare Zustände eines initialen DFAs offensichtlich keine Konsequenzen für die von ihm akzeptierte Sprache haben. Der Reduktionsalgorithmus liefert wiederum einen zusammenhängenden Automaten.}

Beweis von Theorem 4.2.1 ⁺⁺⁺⁾: Sei $A = (S_A, X, Y, n_A, o_A)$ ein endlicher MOORE-Automat; nach Theorem 2.2.1 ⁺⁺⁺⁺⁾ genügt es, die Äquivalenzklassen von A effektiv anzugeben; man konstruiert dann $B = (S_B, X, Y, n_B, o_B)$ wie im Beweis zu Theorem 2.2.1 angegeben; der Anfangszustand von A ist die Äquivalenzklasse, die den Anfangszustand von A enthält, Endzustände von B sind alle Äquivalenzklassen von A , die einen Endzustand von A enthalten. B ist MEALY-äquivalent zu A und MEALY-reduziert.

Der Algorithmus besteht nun darin, daß man der Reihe nach die i -Klassen ⁺⁺⁺⁺⁾ P_i ($i=1, 2, \dots$) von A erzeugt. Da P_{i+1} jeweils eine feinere Klasseneinteilung als P_i ($1 \leq i$) ist und A endlichviele Zustände hat, erhält man nach endlich vielen Schritten die Äquivalenzklassen von A (nach Lemma 4.2.2 ^{+) cf. p.31 ⁺⁺⁾ cf. Def.4.2.5, p.38 ⁺⁺⁺⁾ cf. GLUSCHKOW, p.81 ⁺⁺⁺⁺⁾ cf. p.15 ff. ⁺⁺⁺⁺⁾ cf. Def.4.2.6, p.38}

g.d.w. $P_{i+1}=P_i$ für ein $i, i \geq 1$).

Für das Folgende stellen wir uns die Funktionen n_A und o_A jeweils durch eine Matrix gegeben vor. Die Zerlegung der Zustandsmenge S_A in die 1-Klassen kann man aus der o_A -Matrix sofort ablesen; man⁺ nämlich jeweils Zustände s, t in der gleichen 1-Klasse zusammen, für die gilt: $o_A(s, x) = o_A(t, x)$ für bel. $x \in X$. Dann bildet man die sogenannte 1-Tabelle, indem man in der n_A -Matrix alle Eintragungen von Zuständen durch Eintragungen der 1-Klassen ersetzt, in denen sie enthalten sind. In die gleiche 2-Klasse werden nun alle Zustände aufgenommen, die in der 1-Tabelle eine gleiche Zeile von 1-Klassen aufweisen. Wenn man nun in der n_A -Matrix jeden Zustand durch die 2-Klasse ersetzt, in der er enthalten ist, erhalten wir die 2-Tabelle. Indem wir so fortfahren, erhalten wir schließlich eine $k+1$ -Tabelle, die mit der k -Tabelle übereinstimmt. Damit haben wir die gesuchte Klasseneinteilung, d.h. die Äquivalenzklassen⁺⁺ von A .

Man überzeugt sich leicht, daß ein MOORE-Automat bei dieser Minimierung nur dann ein MOORE-Automat bleibt, wenn je zwei gleichmarkierte Zustände in die gleiche Äquivalenzklasse kommen. Man kann dies dadurch erreichen, daß man zu Beginn der Rechnung gleichmarkierte Zustände in die gleiche 0-Klasse nimmt; zwei 0-äquivalente Zustände heißen 1-äquivalent, wenn sie durch bel. Eingabesymbole wiederum in 0-äquivalente Zustände überführt werden. 1-äquivalente Zustände kommen in die gleiche 1-Klasse, und dann kann das beschriebene Verfahren angewandt werden. Im Anwendungsteil wird ein Beispiel für diesen Algorithmus gegeben. Damit ist Theorem 4.2.1 gezeigt.

N.B.: In der linguistischen Literatur wird oft die Forderung erhoben, eine Grammatik solle einfach sein. Die Einfachheit einer Theorie ist unter Umständen ein schwierig zu präzisierender Begriff. In der Theorie der DFAen bietet sich nun an, von verschiedenen DFAen denjenigen einfacher zu nennen, der weniger Zustände als der andere hat. In der Klasse der zu einem DFA äquivalenten Akzeptoren sind demnach die reduzierten DFAen die einfachsten.

⁺) 'faßt' einfügen!

⁺⁺) cf. p.16

4.3 Finite-State-Sprachen

Def.4.3.1 Eine Sprache L über V heißt Finite-State-Sprache (FSS), wenn es einen DFA $A=(S,V,n,s_0,F)$ gibt mit $L=L(A)$.

Sei $FS(V)$ die Klasse aller FSSen über V .

Wir können für das Folgende o.E.d.A. voraussetzen, daß n stets für alle Zustände von A und alle Symbole von V definiert ist. Wenn dies nämlich nicht der Fall ist, so müssen wir die Definition von n nur unwesentlich verändern. Sei z.B. $n(s,x)$ nicht definiert. Dann setzen wir fest: $n(s,x)=t_A$ wobei t_A ein neuer Zustand ist, der nicht in S liegt und für alle $x \in V$ gilt: $n(t_A,x)=t_A$. Offensichtlich hat diese Vereinbarung keine Konsequenzen für $L(A)$.

FSAen sind offensichtlich Analysegrammatiken.⁺⁾ Dennoch kann man durch sie nicht alle entscheidbaren Sprachen beschreiben. Dies zeigen wir, indem wir eine entscheidbare Sprache L angeben, die durch keinen FSA beschreibbar ist. Später werden wir eine Analysegrammatik für L angeben. ⁺⁺⁾

Satz 4.3.1 Die Sprache $L=\{a^n b^n : n \geq 1\}$ ist durch keinen DFA beschreibbar. (a^n steht für die n -fache Anneinanderreihung von a)

Beweis: Angenommen, es gibt einen DFA $A=(S, \{a,b\}, n, s_0, F)$ mit $L(A)=L$. Wir betrachten nun die Zustandsfolge $M(s_0, a), M(s_0, a^2), \dots, M(s_0, a^n), \dots$. Weil S endlich ist, können diese Zustände nicht alle verschieden sein, d.h. es gibt von einander verschiedene k, l mit: $M(s_0, a^k) = M(s_0, a^l)$, also $M(s_0, a^k b^k) = M(M(s_0, a^k), b^k) = M(M(s_0, a^l), b^k) = M(s_0, a^l b^k)$ ⁺⁺⁺⁾
Nach Vor. gilt: $M(s_0, a^k b^k) \in F$, also $M(s_0, a^k b^l) \in F$, d.h. $a^k b^l \in L$ für ein $k \neq l$. Dies widerspricht der Definition von L . Mithin ist die Annahme widerlegt und so der Satz bewiesen.

⁺⁾ cf. Def.1.2.7, p.4

⁺⁺⁾ cf. den Beweis von Satz 7.3.3

⁺⁺⁺⁾ cf. p.27

5. ZUR CHARAKTERISIERUNG VON FINITE-STATE-SPRACHEN

In diesem Abschnitt werden einige Eigenschaften von FSSen untersucht, durch die sie formal charakterisiert werden können.⁺⁾

5.1 Das Analyse-Synthese-Theorem

Ein FA kann ein sehr kompliziertes Gebilde sein. Man sieht ihm nicht ohne weiteres an, wie die Ketten aussehen, die er akzeptiert. Wir führen nun den Formalismus der sogenannten regulären Sprachen ein; er erlaubt es, Mengen von Ketten in übersichtlicher Form zu schreiben. Der Inhalt des in diesem Abschnitt zu beweisenden Analyse-Synthese-Theorems (AST) ist es, daß man mit Hilfe dieses Formalismus genau die FSSen beschreiben kann.

Def.5.1.0 Seien A, B bel. Sprachen; dann heißt

- 1) $(AB) := \{xy: x \in A \text{ und } y \in B\}$ Produkt von A mit B,
- 2) $(A \cup B) := \{x: x \in A \text{ oder } x \in B\}$ Vereinigung oder Disjunktion von A und B und
- 3) $(A)^* := \bigcup_{i=0} (A)^i$, wobei $(A)^0 = \{e\}$ und $(A)^{i+1} = (A)^i(A)$ für alle $i, i \geq 0$, Sternprodukt von A.

Def.5.1.1 Induktive Definition der regulären Sprachen(über V):

Sei V ein endl. Vokabular. Dann ist:

- 1) a) $\emptyset$ eine reguläre Sprache,
b) $\{e\}$ eine reguläre Sprache und
c) $\{a\}$ eine reguläre Sprache für alle $a \in V$.
- 2) Wenn R_1 und R_2 reguläre Sprachen sind, so sind ebenfalls $(R_1 R_2)$, $(R_1 \cup R_2)$ und $(R_1)^*$ reguläre Sprachen.

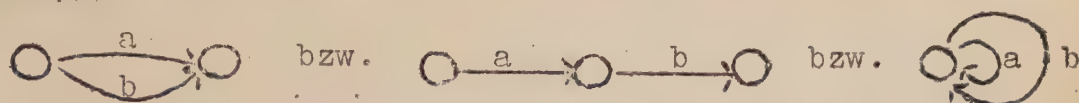
Die Klasse aller regulären Sprachen über V nennen wir $RG(V)$.

Um die Notation von regulären Sprachen übersichtlicher zu machen legen wir fest, daß das Produkt stärker binden soll als die Disjunktion; wir können so Klammern fortlassen.

Außenklammern werden ebenfalls weggelassen. Statt $\{a\}$ bzw. $\{e\}$ schreiben wir auch $\underline{a}$ bzw. $\underline{e}$ und, falls Konfusionen ausgeschlossen sind, einfach a bzw. e.

⁺⁾ 'charakterisieren' ist hier natürlich in einem anderen Sinn zu verstehen als in Abschnitt 3; cf. p.22 f.

Für Elemente $a, b \in V$ ist anschaulich klar, daß den regulären Sprachen $a \cup b$ bzw. ab bzw. ab^* die Graphenschemata



entsprechen. Eine reguläre Sprache kann so als Analyse eines komplexen Graphen im Hinblick auf die Bausteine, aus denen er zusammengesetzt ist (nicht weiter sinnvoll zerlegbare Teilgraphen), aufgefaßt werden. Die Umwandlung einer durch einen FA gegebenen Sprache in eine durch eine reguläre Notation gegebene Sprache nennen wir deshalb Analyse, den umgekehrten Vorgang, d.h. die Konstruktion eines FAs nach einer vorgegebenen regulären Sprache, nennen wir Synthese.

Dieser Sachverhalt liegt dem folgenden Theorem zugrunde:

Theorem 5.1.1 (AST) $FS(V) = RG(V)$

Nach dem Gesagten nennen wir die Behauptung $FS(V) \subseteq RG(V)$ Analysetheorem und die Behauptung $RG(V) \subseteq FS(V)$ Synthesetheorem.

Beweis des Analysetheorems:⁺

Sei $A = (S, V, n, s_0, F)$ ein bel. DFA. Wir geben einen Algorithmus an, mit dessen Hilfe wir eine reguläre Sprache konstruieren können, die gleich $L(A)$ ist. Dazu genügt es, einen Algorithmus⁺⁺ anzugeben, die gleich der Sprache ist, die in A durch irgendeinen Zustand dargestellt wird.⁺⁺⁺

Man erhält so insbesondere die regulären Sprachen, die gleich den durch die Endzustände von A in A dargestellten Sprachen sind, und zwar sind dies endlich viele. Seien diese regulären Sprachen etwa $R_1, \dots, R_k$. Offensichtlich gilt dann: $L(A) = R_1 \cup R_2 \dots \cup R_k$. $L(A)$ ist dann durch eine reguläre Darstellung gegeben und mithin regulär.

⁺) cf. GLUSCHKOW, p.35 ff.

⁺⁺) 'zur Konstruktion einer regulären Sprache' einfügen!

⁺⁺⁺) 'darstellen' wird hier synonym zu 'festlegen' im Sinne von Def. 3.3.2, p.28, benutzt.

Wir denken uns die Zustände von A mit den natürlichen Zahlen $1, \dots, n$ durchnummeriert. Seien i und j ($i, j = 1, \dots, n$) bel. Zustände von A und sei $p = x_1 \dots x_k \in F(V)$ ein Wort, das A aus i in j überführt⁺), d.h. $j = M(i, p)$. Wir sagen dann: A geht von i in j über durch die Zwischenzustände $M(i, x_1), M(i, x_1 x_2), \dots, M(i, x_1 \dots x_{k-1})$. Mit S_{ij}^k ($i, j = 1, 2, \dots, n; k = 0, 1, \dots, n$) bezeichnen wir nun die Sprache, die genau die nichtleeren Ketten über V enthält, die A aus i in j überführen, wobei als Zwischenzustände höchstens die Zustände $1, \dots, k$ vorkommen. S_{ij}^0 bezeichnet die Sprache, die genau die nichtleeren Ketten über V enthält, die A ohne Zwischenzustände von i in j überführen. Wir beweisen nun, daß alle diese S_{ij}^k durch eine reguläre Sprache gegeben werden können, die sich durch einen Algorithmus gewinnen läßt. Damit ist der Beweis des Analysetheorems sofort vollständig, denn falls $i s_0$ bezeichnet und j bel. ist, so ist die in A durch j dargestellte Sprache gleich

$$S_{ij}^n, \text{ falls } i \neq j, \\ S_{ij}^n \cup \underline{e}, \text{ falls } i = j.$$

Wir zeigen also Beh.1: Für alle $k, 0 \leq k \leq n$, bel. $i, j, 1 \leq i, j \leq n$: S_{ij}^k ist regulär.

Bew. durch Ind. über k:

I.A.: $k=0$:

1. Fall: Es gibt kein $x \in V$: $M(i, x) = j$;

dann gilt: $S_{ij}^0 = \emptyset$.

2. Fall: Es gibt genau ein $x \in V$: $M(i, x) = j$;

dann gilt: $S_{ij}^0 = \underline{x}$.

3. Fall: Die Symbole $x_1, \dots, x_1$ überführen A aus i in j ;

dann gilt: $S_{ij}^0 = \underline{x_1 x_2 \dots x_1}$.

In allen drei Fällen ist S_{ij}^0 regulär.

I.S.:

I.V.: $S_{ij}^0, S_{ij}^1, \dots, S_{ij}^k$ sind reguläre Sprachen.

I.B.: S_{ij}^{k+1} ist regulär.

⁺) 'überführen' ist hier genau in dem Sinne zu verstehen, wie es für Automaten eingeführt wurde; cf. Def.3.3.1, p.27.

Dazu zeigen wir

Beh.2: $S_{i \ j}^{k+1} = S_{i \ j}^k \cup S_{i \ k+1}^k (S_{k+1 \ k+1}^k)^* S_{k+1 \ j}^k$

Wir nennen die rechts vom Gleichheitszeichen stehende Sprache R. Unter Benutzung der I.V. sieht man sofort, daß R regulär ist. Nun zeigen wir Beh.2.

Wir zeigen daß $S_{i \ j}^{k+1} \supseteq R$.

In R sind nur solche Ketten, die A aus i in j überführen, wobei als Zwischenzustände höchstens $1, \dots, k$ vorkommen.

Diese Ketten sind natürlich auch in $S_{i \ j}^{k+1}$ enthalten.

Nun zeigen wir, daß $S_{i \ j}^{k+1} \subseteq R$.

Sei p ein bel. Wort aus $S_{i \ j}^{k+1}$.

1.Fall: Bei der Überführung von A aus i in j durch p kommt k+1 nicht als Zwischenzustand vor. Dann gilt:
 $p \in S_{i \ j}^k \subseteq R$.

2.Fall: Bei der Überführung kommt k+1 vor. Man sieht nun leicht ein, daß sich p in der Form $p = p_1 p_2 \dots p_m$ ($m \geq 2$) darstellen läßt, wobei p_1 A aus i in k+1 unter alleiniger Benutzung der Zwischenzustände $1, \dots, k$ überführt, $p_2, \dots, p_{m-1}$ A jeweils aus k+1 in k+1 unter alleiniger Benutzung von $1, \dots, k$ überführen und p_m A aus k+1 unter alleiniger Benutzung von $1, \dots, k$ in j überführt. Also gilt:
 $p_1 \in S_{i \ k+1}^k, p_2 \in S_{k+1 \ k+1}^k, \dots, p_{m-1} \in S_{k+1 \ k+1}^k, p_m \in S_{k+1 \ j}^k$.
 Aus der Definition des Produktes und des Sternproduktes folgt dann sofort: $p \in S_{i \ k+1}^k (S_{k+1 \ k+1}^k)^* S_{k+1 \ j}^k \subseteq R$.

Falls $m=2$, also $p = p_1 p_2$, wobei p_1 A aus i in k+1, p_2 A aus k+1 in j unter alleiniger Benutzung von $1, \dots, k$ überführen, ist auszunutzen, daß $e \in (S_{k+1 \ k+1}^k)^*$,
 also $p_1 p_2 = p_1 e p_2 \in S_{i \ k+1}^k (S_{k+1 \ k+1}^k)^* S_{k+1 \ j}^k \subseteq R$.

In beiden Fällen ist also $p \in R$. Damit ist Beh.2 bewiesen. Zusammen mit der I.V. liefert die in Beh.2 gebrachte Formel einen für den Beweis von Beh.2 gewünschten Algorithmus, wie wir uns an dem nachfolgenden Beispiel klarmachen können. Damit ist das Analysetheorem bewiesen.

Wir verdeutlichen uns die durch das Analysetheorem gelieferte Konstruktion nun an einem Beispiel.

Sei $A = (\{1, 2\}, \{a, b\}, n, \{1\}, \{1, 2\})$ ein DFA. n sei durch die folgende Matrix gegeben:

	a	b
1	1	2
2	1	2

Offenbar gilt dann:

$$L(A) = S_{1,1}^2 \cup S_{1,2}^2 \cup \underline{e} \quad .+)$$

Nach Beh. 2 gilt:

$$S_{1,1}^2 = S_{1,1}^1 \cup S_{1,1}^1 (S_{1,1}^1)^* S_{1,1}^1 \text{ und } S_{1,2}^2 = S_{1,2}^1 \cup S_{1,1}^1 (S_{1,1}^1)^* S_{1,2}^1,$$

$$S_{1,1}^1 = S_{1,1}^0 \cup S_{1,1}^0 (S_{1,1}^0)^* S_{1,1}^0 \text{ und } S_{1,2}^1 = S_{1,2}^0 \cup S_{1,1}^0 (S_{1,1}^0)^* S_{1,2}^0.$$

Wie man nun aus der n -Matrix ablesen kann gilt:

$$S_{1,1}^0 = \underline{a} \text{ und } S_{1,2}^0 = \underline{b}.$$

Also ist

$$S_{1,1}^2 = (\underline{a} \cup \underline{a}(\underline{a})^* \underline{a}) \cup (\underline{a} \cup \underline{a}(\underline{a})^* \underline{a})(\underline{a} \cup \underline{a}(\underline{a})^* \underline{a}) \cup (\underline{a} \cup \underline{a}(\underline{a})^* \underline{a})$$

$$S_{1,2}^2 = (\underline{b} \cup \underline{a}(\underline{a})^* \underline{b}) \cup (\underline{a} \cup \underline{a}(\underline{a})^* \underline{a})(\underline{a} \cup \underline{a}(\underline{a})^* \underline{a}) \cup (\underline{b} \cup \underline{a}(\underline{a})^* \underline{b})$$

Durch Einsetzung in den obigen Ausdruck für $L(A)$ erhalten wir so eine ziemlich komplizierte reguläre Darstellung von $L(A)$. Diese Darstellung ließe sich mit Hilfe der folgenden wichtigsten Rechenregeln für reguläre Sprachen, deren Richtigkeit man leicht einsieht, u. U. noch vereinfachen.

$$1) (R_1 \cup R_2) \cup R_3 = R_1 \cup (R_2 \cup R_3)$$

$$2) R_1 \cup R_2 = R_2 \cup R_1$$

$$3) (R_1 R_2) R_3 = R_1 (R_2 R_3)$$

$$4) (R_1 \cup R_2) R_3 = R_1 R_3 \cup R_2 R_3, \quad R_1 (R_2 \cup R_3) = R_1 R_2 \cup R_1 R_3$$

$$5) (R^*)^* = R^*$$

$$6) R^* = \underline{e} \cup R R^*$$

$$7) R R^* = R^* R$$

$$8) \underline{e}^* = \underline{e}$$

$$9) R \underline{e} = \underline{e} R = R$$

$$10) (R_1^* R_2^*)^* = (R_1 \cup R_2)^*$$

$$11) R \emptyset = \emptyset R = \emptyset$$

$$12) R \cup \emptyset = R$$

$$13) \emptyset^* = \underline{e}$$

+) Daß $\underline{e} \in L(A)$ ergibt sich aus der Konvention von p. 31

Beweis des Synthesetheorems: +)

Wir zeigen, daß es für eine bel. reguläre Sprache R einen FA gibt, der R akzeptiert. Meist wird ein derartiger FA ein NFA sein, zu dem es aber nach Theorem 4.1.1 einen äquivalenten DFA gibt. Da wir im folgenden den Begriff der Überführbarkeit ⁺⁺) auch für NFAen brauchen, legen wir fest:

Def.5.1.2 Sei $A=(S,V,n,s_0,F)$ ein NFA, p ein bel. Wort aus $F(V)$, $p=x_1 \dots x_k$. Dann sagen wir:
A wird durch p aus dem Zustand s in den Zustand t überführt, wenn es eine Zustandsfolge $s_0, \dots, s_k$ gibt, so daß gilt:

- 1) $s_0=s$,
- 2) $s_i \in n(s_{i-1}, x_i)$ für alle $i, 0 < i \leq k$, und
- 3) $s_k=t$.

Wir zeigen nun das Synthesetheorem durch Induktion über die regulären Sprachen.

I.A.:

$\emptyset$, ϵ und a für alle $a \in V$ sind offenbar in $FS(V)$.

I.S.:

1) Disjunktion

I.V.: Es gibt DFAen $A=(S_A, V, n_A, s_0^A, F_A)$, $B=(S_B, V, n_B, s_0^B, F_B)$ mit $L=L(A)$, $M=L(B)$ für bel. Sprachen aus $RG(V)$.

I.B.: Es gibt einen FA C mit $L(C)=L \cup M$.

Def.: $C:=(S_A \boxtimes S_B, V, n_C, (s_0^A, s_0^B), (F_A \boxtimes S_B) \cup (S_A \boxtimes F_B))$,
wobei für alle $s \in S_A$, $t \in S_B$, $x \in V$ gelte:
 $n_C((s,t), x) := (n_A(s, x), n_B(s, x))$

Beh.: $L(C)=L \cup M$

Bew.: Sei p ein bel. Wort aus $F(V)$:

$p \in L(C)$

g.d.w. $M_C((s_0^A, s_0^B), p) \in (F_A \boxtimes S_B) \cup (S_A \boxtimes F_B)$

g.d.w. $(M_A(s_0^A, p), M_B(s_0^B, p)) \in F_A \boxtimes S_B$ oder

$(M_A(s_0^A, p), M_B(s_0^B, p)) \in S_A \boxtimes F_B$ (nach Def. von n_C)

g.d.w. $(M_A(s_0^A, p) \in F_A \text{ und } M_B(s_0^B, p) \in S_B)$ oder

stets erfüllt!

$(M_A(s_0^A, p) \in S_A \text{ und } M_B(s_0^B, p) \in F_B)$

stets erfüllt!

g.d.w. $M_A(s_0^A, p) \in F_A$ oder $M_B(s_0^B, p) \in F_B$ g.d.w. $p \in L(A) \cup L(B)$

+) cf. GINSBURG 1962, 115 f.; cf. auch p.43 ⁺⁺) cf. Def.3.3.1, p.27

Damit ist die I.B. für die Disjunktion bewiesen. Als nächstes behandeln wir den Fall des Produktes.

2) Produkt

I.V.: Es gibt DFAen $A=(S_A, V, n_A, s_0^A, F_A)$, $B=(S_B, V, n_B, s_0^B, F_B)$ mit $L(A)=L$, $L(B)$ für bel. $L, M \in RG(V)$. Wir können o.E.d.A. voraussetzen: $S_A \cap S_B = \emptyset$.

I.B.: Es gibt einen FA C mit $L(C)=LM$.

Bew.: Wir definieren einen NFA^{+} C und zeigen: $L(C)=LM$.
 $C:=(S_A \cup S_B, V, n_C, \{s_0^A\}, F_B)$, wobei für bel. $s \in S_A \cup S_B$, $x \in V$ gelte:

$$n_C(s, x) := \begin{cases} \{n_A(s, x)\} & , \text{ falls } s \in S_A - F_A \\ \{n_A(s, x), n_B(s_0^B, x)\} & , \text{ falls } s \in F_A \\ \{n_B(s, x)\} & , \text{ falls } s \in S_B \end{cases}$$

Wir bemerken, daß die zweite Zeile der Definition von n_C die beiden FSAen A und B "koppelt". Ein bel. Wort $p \in L(A)$ überführt C aus s_0^A in einen Zustand aus F_A oder in den Zustand s_0^B . Somit gilt für bel. $p \in F(V)$:

$p \in LM$ g.d.w. $p=p_1p_2$ mit $p_1 \in L$ und $p_2 \in M$

g.d.w. $M_A(s_0^A, p_1) \in F_A$ und $M_B(s_0^B, p_2) \in F_B$

g.d.w. $p_1p_2 \in L(C)$

(C wird nämlich durch p_1 aus s_0^A in s_0^B überführt, aus s_0^B dann durch p_2 in einen Zustand in F_B^{++})

Damit ist der Produktfall erledigt..

3) Sternprodukt

I.V.: Es gibt einen DFA $A=(S, V, n_A, s_0^A, F_A)$ mit $L=L(A)$ für eine bel. Sprache $L \in RG(V)$.

I.B.: Es gibt einen FA B mit $L(B)=L^*$.

Bew.: Wir definieren einen NFA^{+} B und zeigen, daß $L=L(B)$.

$B:=(S, V, n_B, \{s_0^A\}, \{F_A, s_0^A\})$, wobei für bel. $s \in S$, $x \in V$ gelte:

$$n_B(s, x) := \begin{cases} \{n_A(s, x)\} & , \text{ falls } s \in S - F \\ \{n_A(s, x), n_A(s_0^A, x)\} & , \text{ falls } s \in F \end{cases}$$

Durch die zweite Zeile der Definition von n_B wird sichergestellt, daß B durch ein bel. Wort aus L aus s_0^A u.a. wieder

(+) cf. p.47 oben

++) 'überführen' ist hier und im Fall des Sternproduktes im Sinne von Def.5.1.2, p.47, zu verstehen.

nach s_0^A überführt wird.⁺) Nun zeigen wir die I.B..

Wir zeigen zuerst: a) $L^* \subseteq L(B)$

Zum Beweis von (a) genügt es, für alle nat. Zahlen i zu zeigen: $L^i \subseteq L(B)$, denn jedes $p \in L^*$ liegt in einem L^i .

Bew. von (a) durch Induktion über i :

I.A.(a): $i=0$:

Wir müssen zeigen, daß $L^0 = e \subseteq L(B)$. Weil die Mengen der Anfangs- und Endzustände von B nicht disjunkt sind, so $e \in L(B)$.⁺⁺)

I.S.(a):

I.V.(a): $L^i \subseteq L(B)$

I.B.(a): $L^{i+1} = L^i L \subseteq L(B)$

Nach I.V. gilt: $L = L(A)$. Es gilt ferner für bel. $p \in F(V)$:

$p \in L^i L$ g.d.w. $p = p_1 p_2$ mit $p_1 \in L^i$ und $p_2 \in L(A)$

g.d.w. $p_1 \in L^i$ und $M_A(s_0^A, p_2) \in F_A$

g.d.w. $p_1 p_2 \in L(B)$

(Nach I.V.(a) wird B durch p_1 aus s_0^A in einen Endzustand überführt⁺). Nach Definition von n_B wird B dann auch wieder nach s_0^A aus s_0^A durch p_1 überführt und durch p_2 aus s_0^A in einen Endzustand von B überführt.) Damit ist (a) bewiesen.

Nun zeigen wir: b) $L(B) \subseteq L^*$.

Sei $p = x_1 \dots x_k$ ein bel. nichtleeres (keine Einschränkung!)

Wort von $L(B)$. Wir zeigen: $p \in L^*$.

Offensichtlich läßt sich p dann in der Form $p = p_1 \dots p_m$

$(1 \leq m \leq k)$ darstellen, wobei $(1 \leq i < m)$ B aus

s_0^A in s_0^A überführt und p_m B aus s_0 in einen Endzustand

von F_A überführt. Falls $m=1$, so wird B durch p direkt aus s_0^A in einen Endzustand⁺⁺⁺) überführt. Auf Grund der

Konstruktion von B und der I.V. gilt dann für alle

$i, 1 \leq i \leq m$: $M_A(s_0^A, p_i) \in F$. Mithin sind alle diese

$p_i \in L$, also $p_1 \dots p_m \in L^m \subseteq L^*$.

Damit ist der Fall des Sternprodukts erledigt. Also

ist das Synthesetheorem vollständig bewiesen und somit auch das AST.

⁺) cf. (++) , p.48

⁺⁺) Diesen Fall hat GINSBURG 1962, p.116 bei seiner Konstruktion übersehen.

⁺⁺⁺) 'in F_A ' einfügen!

N.B. Die Bedeutung des AST für die Anwendung der Theorie der FSSen muß hoch eingeschätzt werden. Das AST liefert einen handlichen Formalismus zur Beschreibung von FSSen. Es erlaubt auch, die Leistungsfähigkeit traditioneller linguistischer Theorien an Begriffen der Theorie der FAen zu messen; so scheinen z.B. einige Notationen der Tagmiker oder etwa HJELMSLEVs den Notationen für reguläre Sprachen äquivalent zu sein. Auch dürfte das AST bei einer Automatisierung von discovery procedures eine wichtige Rolle spielen; dies hat der Verfasser in einem Seminarvortrag WS 1967/68 darzulegen versucht.

5.2 Die Charakterisierungssätze von MYHILL und NERODE

Neben dem AST kommt den beiden Sätzen, die in diesem Abschnitt bewiesen werden, eine hervorragende Rolle bei der Charakterisierung der Klasse der FSSen zu. Zur Formulierung der Sätze benötigen wir einige Definitionen.

Def.5.2.1 Eine Äquivalenzrelation R über $F(V)$ ist rechtsinvariant (linksinvariant) in (bezug auf die Verkettung) g.d.w. für bel. $z \in F(V)$ gilt: wenn xRy so $xzRyz$ (wenn xRy so $zxRzy$).

Eine rechts- und linksinvariante Äquivalenzrelation heißt Kongruenzrelation.

Def.5.2.2 Sei R eine Äquivalenzrelation. Die Klasse aller zu p äquivalenten Ketten $\bar{p} := \{p' : pRp'\}$ ist eine der von R erzeugten Äquivalenzklassen.

Eine Äquivalenzrelation hat einen endlichen Index g.d.w. sie endlichviele Äquivalenzklassen erzeugt.

Def.5.2.3 Seien p_1 bzw. q_1 Elemente der Äquivalenzklasse $\bar{p}$ bzw. $\bar{q}$; dann ist das Produkt der beiden Äquivalenzklassen eindeutig definiert durch $\bar{p} \bar{q} := \overline{p_1 q_1}$.

Nun können wir den folgenden Satz formulieren:

Satz 5.2.1 (MYHILL) Die folgenden drei Aussagen über eine Sprache L über V sind äquivalent:

- 1) $L \in FS(V)$
- 2) L ist die Vereinigung einiger Äquivalenzklassen einer Kongruenzrelation von endlichem Index.
- 3) Sei für alle $z, w \in F(V)$: pEq g.d.w. ($zpw \in L$ g.d.w. $zqw \in L$). Dann ist E eine Kongruenzrelation von endlichem Index.

Beweis:⁺) Wir müssen zeigen: (1) impliziert (2),
(2) impliziert (3) und (3) impliziert (1).

a) (1) impliziert (2)

Bew. von (a): Nach Vor. gibt es einen DFA $A=(S,V,n,s_0,F)$ mit $L(A)=L$. Wir definieren eine Kongruenzrelation K und zeigen, daß K einen endlichen Index hat, ferner, daß L die Vereinigung einiger von K erzeugter Äquivalenzklassen ist.

Def.: pKq g.d.w. für alle $s \in S$: $M(s,p)=M(s,q)$.

Beh.aa: K ist ein Kongruenzrelation von endlichem Index.

K ist offensichtlich eine Äquivalenzrelation. Wir zeigen, daß K rechtsinvariant ist; wir setzen also voraus: pKq ; dies ist genau dann der Fall, wenn für bel.

$s \in S$ gilt: $M(s,p)=M(s,q)$. Nun gilt für ein bel. $z \in F(V)$:
 $M(s,pz)=M(M(s,p),z)$ ⁺⁺⁾
 $=M(M(s,q),z)$
 $=M(s,qz)$

K ist also rechtsinvariant. Ganz analog zeigt man die Linksinvarianz von K . K ist also eine Kongruenzrelation.

Wir zeigen nun, daß K einen endlichen Index hat. Dazu betrachten wir die von K erzeugten Äquivalenzklassen.

Sei p ein bel. Wort aus $F(V)$, dann gilt:

$\bar{p} = \{ p' : M(s,p)=M(s,p') \text{ für alle } s \in S \}$.

Nun definieren wir für alle $p \in F(V)$, $s \in S$:

$M_p(s) := M(s,p)$;

M_p ist also eine Abbildung von S in S für ein bel. festes p . Dann gilt für alle $p \in F(V)$:

$\bar{p} = \{ p' : M_p(s)=M_{p'}(s) \text{ für alle } s \in S \}$.

Sei nun r die Anzahl der Zustände von A ; dann gibt es höchstens r^r verschiedene Abbildungen von S in S .

Ganz unabhängig von der Wahl von p ist dann M_p eine dieser r^r Abbildungen. Abbildungen und Äquivalenz-

klassen sind einander eineindeutig zugeordnet. K kann deshalb auch nur höchstens r^r verschiedene Äquivalenzklassen erzeugen. Damit ist Beh.aa bewiesen.

Beh.ab: L ist die Vereinigung einiger von K erzeugter Äquivalenzklassen.

⁺) cf. RABIN-SCOTT, p.117 f.

⁺⁺) cf. p.27

Bew. von Beh.ab: Sei p ein bel. Element von $L(A)$; dann gilt für alle Ketten p' aus $\bar{p}$: $M(s_0, p) = M(s_0, p') \in F$; d.h. alle Elemente aus $\bar{p}$ sind in $L(A)$. Dann gilt:

$$L = L(A) = \bigcup_{p \in L(A)} \bar{p}.$$

Man sondert also aus den endlich vielen von K erzeugten Äquivalenzklassen diejenigen aus, die einen Satz von L enthalten. (Es gibt nur endlichviele solcher Klassen.) L ist also die Vereinigung einiger durch K erzeugter Äquivalenzklassen. Damit ist Beh.ab bewiesen und somit auch (a).

b) (2) impliziert (3)

Bew. von (b): Sei C eine Kongruenzrelation über $F(V)$ mit endlichem Index, L die Vereinigung einiger durch C erzeugter Äquivalenzklassen. Sei E wie in (3) definiert. Offensichtlich ist E eine Kongruenzrelation.

Beh.: E hat einen endlichen Index.

Wir zeigen, daß E die größte Kongruenzrelation über $F(V)$ ist, daß also jede durch C erzeugte Äquivalenzklasse in einer durch E erzeugten Äquivalenzklasse liegt. Wegen der Disjunktheit der Äquivalenzklassen kann dann E höchstens die gleiche Anzahl von Äquivalenzklassen wie C erzeugen, E hat also erst recht einen endlichen Index.

Seien p, q bel. Elemente aus $F(V)$ mit pCq . Sei nun $zpw \in L$. Dann ist zunächst $zpwCzqw$, weil C eine Kongruenzrelation ist; weil aber nach Vor. L die Vereinigung einiger durch C erzeugter Äquivalenzklassen ist, so gilt: $zqw \in L$. Analog zeigt man, daß $zqw \in L$ zur Folge hat, daß $zpw \in L$. Somit folgt aus pCq , daß pEq . Damit ist die Beh. und folglich (b) bewiesen. (Daß E die größte Äquivalenzrelation über $F(V)$ ist, ergibt sich aus der bel. Wahl von C .)

c) (3) impliziert (1)

Bew. von (c): Sei E die in (3) definierte Kongruenzrelation von endlichem Index. Wir definieren einen DFA A mit $L(A) = L$.

Def.: $A := (S, V, n, s_0, F)$, wobei S die Menge der durch E erzeugten Äquivalenzklassen ist; S ist nach Vor. endlich.

Sei nun $\bar{p}$ eine bel. Äquivalenzklasse aus S , q ein bel. Element aus $\bar{p}$; dann definieren wir für alle $x \in V$:

$$n(\bar{p}, x) := \overline{qx},$$

Sei nun r ein bel. Element aus $\bar{p}$ und $r \neq q$. Dann gilt: qEr . Weil E rechtsinvariant ist, so $qxEr$; also $\overline{qx} = \overline{rx}$. n ist also unabhängig von der Auswahl eines bestimmten Elementes von $\bar{p}$ definiert.

Sei $s_0 := \bar{e}$ und $F := \{\bar{p} : p \in L\}$. Damit ist A vollständig definiert.

Beh.: $L = L(A)$.

Bew. der Beh.: Zunächst gilt wieder: L ist die Vereinigung einiger durch E erzeugter Äquivalenzklassen. Sei nämlich p ein bel. Element aus L und pEq ; dies genau dann der Fall, wenn für bel. $z, w \in F(V)$ gilt: $zpw \in L$ g.d.w. $zqw \in L$; insbesondere gilt dann: $epe \in L$ g.d.w. $eqe \in L$. Also gilt: $p \in L$ g.d.w. $q \in L$. Also $L = \bigcup_{p \in L} \bar{p}$.

Sei nun p ein bel. Element aus L . Wir zeigen, daß genau in diesem Fall $p \in L(A)$, daß also gilt: $M(s_0, p) \in F$.

Zunächst ist nach Def. von A $\bar{p} \in F$. p habe die Darstellung $x_1 \dots x_k$. Nun gilt:

$$\begin{aligned} M(s_0, p) &= M(\bar{e}, x_1 \dots x_k) \\ &= M(n(\bar{e}, x_1), x_2 \dots x_k) = M(\overline{ex_1}, x_2 \dots x_k) = M(\bar{x}_1, x_2 \dots x_k) \\ &= M(n(\bar{x}_1, x_2), x_3 \dots x_k) = M(\overline{x_1 x_2}, x_3 \dots x_k) \\ &\vdots \\ &= M(n(\overline{x_1 \dots x_{k-2}}, x_{k-1}), x_k) = M(\overline{x_1 \dots x_{k-1}}, x_k) \\ &= \overline{x_1 \dots x_k} = \bar{p} \end{aligned}$$

Also gilt für bel. $p \in F(V)$: $M(s_0, p) \in F$ g.d.w. $p \in L$.

Also ist $L = L(A)$. Damit ist (c) bewiesen und somit auch Satz 5.2.1.

N.B. BAR-HILLEL ⁺⁾ bemerkt zu Satz 5.2.1, daß er die Explizierung des Begriffs 'syntaktische Kategorie' nahelegt. Sei nämlich L_V eine FSS; dann gibt es eine Kongruenzrelation R , die L_V in endlich viele Äquivalenzklassen zerlegt. Wir können nun die Äquivalenzklasse, zu der $p \in F(V)$ gehört, die syntaktische Klasse von p nennen. Sei etwa $p_1 p_2 p_3$ ein Satz von L_V . Es ist klar, daß wir

⁺⁾ cf. BAR-HILLEL, p.93

für jedes p_i ($i=1,2,3$) eine R-äquivalente Kette einsetzen können und dabei das Ergebnis der Einsetzungen in jedem Fall ein Satz von L bleibt. Diese Eigenschaft der Kommutabilität (Austauschbarkeit) von Ketten in beliebigen Kontexten ohne Einfluß auf die Akzeptierbarkeit eines Satzes ist in der Linguistik oft zur Hilfe genommen worden, um den Begriff der syntaktischen Kategorie zu definieren. Mithin ist der durch die Theorie der FSSen gelieferte Begriff der R-Äquivalenz von Ketten eine brauchbare Explikation dieses wichtigen traditionellen Begriffes.

Der folgende Satz ist eine Verschärfung von Satz 5.2.1.

Satz 5.2.2 (NERODE) Die folgenden drei Aussagen über eine Sprache L über V sind äquivalent.

- 1) $L \in FS(V)$
- 2) L ist die Vereinigung einiger Äquivalenzklassen einer rechtsinvarianten Äquivalenzrelation von endlichem Index.
- 3) Sei für alle $z \in F(V)$: pRq g.d.w. ($pz \in L$ g.d.w. $qz \in L$). Dann ist R eine rechtsinvariante Äquivalenzrelation von endlichem Index.

Der Beweis dieses Satzes ist ganz analog zu dem des vorigen zu führen.⁺) Dabei kann man die in (2) verlangt rechtsinvariante Äquivalenzrelation folgendermaßen definieren:

pQq g.d.w. für bel. $w \in F(V)$: $M(s_0, pw) = M(s_0, qw)$.

Q hat die für den Beweis erforderlichen Eigenschaften.

M ist die verallgemeinerte Überföhrungsfunktion des DFA, der nach (1) L beschreibt.

Das folgende Korollar stellt einen Zusammenhang zwischen Satz 5.2.2 und dem Reduktionstheorem für DFAen ⁺⁺⁾ her.

Korollar 5.2.1 Sei $L \in FS(V)$, R wie in Satz 5.2.2 (3) definiert. Dann ist die Anzahl der durch R erzeugten Äquivalenzklassen gleich der Anzahl der Zustände eines zusammenhängenden MOORE-reduzierten DFA, der L akzeptiert.

Beweis von Korollar 5.2.1: Der Beweis der Behauptung von Satz 5.2.2, daß (3) (1) impliziert, liefert einen DFA A mit $L(A)=L$. Die Zustände von A sind die durch R erzeugten

⁺) cf. RABIN-SCOTT, p.118

⁺⁺⁾ cf. Korollar 4.2.2, p.37

Äquivalenzklassen. $\bar{e}$ ist der Anfangszustand von A und n_A ist, wie in der Konstruktion beschrieben ^{+))}, definiert. Angenommen, A sei nicht MOORE-reduziert. Dann gibt es nach Def.4.2.4 Zustände $\bar{p}, \bar{q}$ ($\bar{p} \neq \bar{q}$) von A, die äquivalent sind, d.h. $M_A(\bar{p}, w) = M_A(\bar{q}, w)$ für bel. $w \in F(V)$. Nach Def. von n_A gilt aber: $M_A(\bar{p}, w) = \overline{pw}$ und $M_A(\bar{q}, w) = \overline{qw}$. Also $\overline{pw} = \overline{qw}$. Dies ist nach Def.5.2.3 genau dann der Fall, wenn $\bar{p} \bar{w} = \bar{q} \bar{w}$, woraus folgt: $\bar{p} = \bar{q}$. Damit ist die Annahme widerlegt und folglich A MOORE-reduziert.

N.B. Ein MOORE-reduzierter DFA A mit $L(A) = L$ liefert so neben dem ^{+++++))} für die Einfachheit von L ^{++))} auch strukturelle Aussagen über L. Die Zustände von A können nach dem Gesagten ^{+++))} nämlich als durch die Relation R ^{++++))} erzeugte syntaktische Klassen aufgefaßt werden. Wie groß der linguistische Aufschlußwert dieser Klasseneinteilung von L ist, bleibe dahingestellt. Immerhin spiegelt sich in ihr die Tatsache, daß eine einmal begonnene Äußerung nicht in beliebiger Weise fortgeführt werden kann, sondern durch das bereits Geäußerte in ihrem Fortgang eingeschränkt sein kann. Jeder Satzanfang ist nämlich Element einer durch R erzeugten Äquivalenzklasse; diese Äquivalenzklasse ist zugleich ein Zustand des Automaten A, der die Möglichkeit, wie dieser Anfang ergänzt werden kann, einschränkt.

6. OPERATIONEN ÜBER FINITE-STATE-SPRACHEN

Im folgenden zeigen wir, daß die Klasse der FSSen gegenüber einigen wichtigen Operationen in jeweils noch zu erläuternder Hinsicht abgeschlossen ist. Es handelt sich um BOOLEsche Operationen, die Substitution und Sequenzentransducer-Abbildungen. Wir bemerken, daß die beiden letzten Abgeschlossenheitseigenschaften auch für die sogenannten kontextfreien Sprachen, von denen noch zu reden sein wird, gelten. ^{+++++))} Für den Beweis der Sätze in diesem Kapitel benutzen wir das AST z.T. wesentlich.

^{+))} cf. p.52 f. ^{++))} cf. p.40 ^{+++))} cf. p.53 f.

^{++++))} R ist wie in Satz 5.2.2 (3) definiert.

^{+++++))} cf. 7.1

^{++++++))} 'Maß' einfügen!

6.1 BOOLEsche Operationen

In diesem Abschnitt zeigen wir die Abgeschlossenheit von $FS(V)$ gegenüber den BOOLEschen Operationen, also:

Satz 6.1.1 Seien $L_1, L_2 \in FS(V)$, dann gilt:

- 1) $L_1 \cup L_2 \in FS(V)$,
- 2) $L_1 \cap L_2 \in FS(V)$ und
- 3) $F(V) - L_1 \in FS(V)$.

Bew. von (1): (1) haben wir beim Beweis des Synthesetheorems bereits bewiesen (Disjunktion). +)

Bew. von (2): Nach Vor. gibt es DFAen $A=(S_A, V, n_A, s_0^A, F_A)$ und $B=(S_B, V, n_B, s_0^B, F_B)$ mit $L_1=L(A)$ und $L_2=L(B)$. Wir definieren $C:=(S_A \boxtimes S_B, n_C, (s_0^A, s_0^B), F_A \boxtimes F_B)$, wobei für alle $s \in S_A$, $t \in S_B$, $x \in V$ gelte: $n_C((s, t), x) := (n_A(s, x), n_B(t, x))$.

Beh.: $L(C) = L_1 \cap L_2$

Bew. der Beh.: Sei p ein bel. Element von $F(V)$:

$p \in L(C)$ g.d.w. $M_C((s_0^A, s_0^B), p) \in F_A \boxtimes F_B$

g.d.w. $(M_A(s_0^A, p), M_B(s_0^B, p)) \in F_A \boxtimes F_B$

g.d.w. $M_A(s_0^A, p) \in F_A$ und $M_B(s_0^B, p) \in F_B$

g.d.w. $p \in L_1 \cap L_2$

Bew. von (3): Nach Vor. gibt es einen DFA $A=(S, V, n, s_0, F)$ mit $L(A)=L_1$. Wir definieren $-A:=(S, V, n, s_0, S-F)$.

Offensichtlich ist $L(-A)=F(V)-L_1$, denn für bel. $p \in F(V)$ gilt: $p \in L(-A)$ g.d.w. $M(s_0, p) \in S-F$ g.d.w. $p \notin L(A)$.

Damit ist Satz 6.1.1 bewiesen.

6.2 Substitution

Die Substitution, mit der wir uns in diesem Abschnitt beschäftigen wollen, ist für die Anwendung eine besonders interessante Operation. ++)

Def.6.2.1 Sei für jedes Element x eines endlichen Vokabulars V V_x ein endliches Vokabular, $f(x) \subseteq F(V_x)$; wir definieren für alle $p \in F(V)$:

$$f(p) := \begin{cases} f(x_1) \dots f(x_k), & \text{falls } p = x_1 \dots x_k \\ \underline{e}, & \text{falls } p = e \end{cases}$$

f ist eine eindeutige Abbildung von $F(V)$ in $P(F(\bigcup_{x \in V} V_x))$. f heißt Substitution.

+) cf. p. 47 ++) cf. besonders 8.1 und 8.2

Falls $f(x) \in RG(V_x)$ für alle $x \in V$, so heißt f reguläre Substitution. ⁺⁺⁾

Sei L eine bel. Sprache über V , f eine Substitution; wir definieren:

$$f^+(L) := \bigcup_{p \in L} f(p)$$

f^+ ist also eine eindeutige Abbildung von $P(F(V))$ in $P(F(\bigcup_{x \in V} V_x))$. f^+ heißt

Substitution (für Sprachen). Analog zu dem oben Gesagten kann f^+ auch als reguläre Substitution verstanden werden, wenn die entsprechenden Bedingungen erfüllt sind.

Nun können wir den folgenden Abgeschlossenheitssatz formulieren:

Satz 6.2.1 Die Klasse der FSSen ist gegenüber der regulären Substitution abgeschlossen.

Genauer formuliert: Wenn $L \in FS(V)$, f eine Substitution ist und $f(x) \in FS(V_x)$ für alle $x \in V$, so $f^+(L) \in FS(\bigcup_{x \in V} V_x)$.

⁺⁾

Beweis: Sei V ein Vokabular mit den Elementen $x_1, \dots, x_n$, $L \in FS(V)$; dann ist nach dem AST $L \in RG(V)$. Dann gibt es eine reguläre Sprache R , so daß $L=R$. Weil für alle $x_i \in V$ nach Vor. $f(x_i) \in FS(V_{x_i})$, so wiederum nach dem AST $f(x_i) \in RG(V_{x_i})$. R wird aus den regulären Sprachen $x_1, \dots, x_n$ durch endlichfache Anwendug der Operationen der Disjunktion, des Produktes und des Sternproduktes gewonnen. Wir können nun alle x_i jeweils durch die regulären Sprachen $f(x_i)$ ersetzen. Die endliche Vereinigung dieser $f(x_i)$ ist dann in $RG(\bigcup_{x \in V} V_x)$.

Nun wenden wir die Operationen der Disjunktion, des Produktes und des Sternproduktes statt auf die x_i auf die $f(x_i)$ in derselben Reihenfolge an, wie zur Gewinnung von R . Wir erhalten so offenbar die Menge $f^+(L)$, die nach dem AST dann in $FS(\bigcup_{x \in V} V_x)$ liegt.

⁺⁾ cf. GINSBURG 1966, p.49

⁺⁺⁾ Statt $f(x) \in RG(V_x)$ können wir wegen dem AST natürlich ebensogut $f(x) \in FS(V_x)$ voraussetzen.

6.3 Sequenzentransducer-Abbildungen

In diesem Abschnitt behandeln wir die Abgeschlossenheit der FSSen gegenüber Sequenzentransducer-Abbildungen. Jede Sequenzentransducer-Abbildung kann als verallgemeinerte reguläre Substitution⁺ aufgefaßt werden, wie sich zeigen wird.

Def.6.3.1 Ein (initialer) Sequenzentransducer ⁺⁺) (ST)

$T=(S,X,Y,R,s_0)$ ist ein 5-Tupel, wobei

- 1) S eine endliche Menge (die Menge der Zustände von T) ist,
- 2) X ein endliches Vokabular (das Eingabealphabet von T) ist,
- 3) Y ein endliches Vokabular (das Ausgabealphabet von T) ist,
- 4) $s_0 \in S$ (der Anfangszustand von T) ist und
- 5) R eine endliche Teilmenge von $S \times F(X) \times F(Y) \times S$ (die Menge der Regeln von T) ist.

Def.6.3.2 Sei $T=(S,X,Y,R,s_0)$ ein ST. Dann nennen wir die durch s_0 festgelegte Transducerabbildung ⁺⁺⁺) T_{s_0} Sequenzentransducer-Abbildung (von T).

Nun können wir den folgenden Satz formulieren:

Satz 6.3.1 Die Klasse der FSSen ist hinsichtlich der Sequenzentransducer-Abbildungen abgeschlossen.

Genauer formuliert: Sei $L_X \in FS(X)$, $T=(S,X,Y,R,s_0)$ ein ST, dann ist $T_{s_0}^*(L_X) = \bigcup_{p \in L_X} T_{s_0}(p) \in FS(Y)$.

Beweis: ⁺⁺⁺⁺) Nach Vor. gibt es einen ST $T=(S,X,Y,R,s_0)$; ferner gilt nach dem AST: $L_X \in RG(X)$, da $L_X \in FS(X)$.

Beh.: $T_{s_0}^*(L_X) \in RG(Y)$.

Aus der Beh. folgt nach dem AST sofort Satz 6.3.1.

⁺) cf. p.56 f.

⁺⁺) cf. p.8, (2) im Anschluß an Def.1.3.5

⁺⁺⁺) cf. Def.1.3.2, p.7

⁺⁺⁺⁺) cf. GINSBURG 1966, 92 f.

Bew. der Beh. Wir geben zunächst eine Übersicht über den Beweisgang. 1) Wir definieren zunächst einen ST T' . 2) Dann definieren wir Mengen M und Q_{L_X} , deren Elemente Ketten sind, die über den Regeln von T' gebildet sind. Wir definieren dann eine reguläre Substitution f (3) und zeigen, daß $T_{s_0}^*(L) = f^+(M \cap Q_{L_X})$ (4). Schließlich zeigen wir noch die Regularität von M (5) und von $M \cap Q_{L_X}$ (6). Auf grund von o Satz 6.2.1 ist dann $f^+(M \cap Q_{L_X})$ regulär und damit auch $T_{s_0}^*(L)$.

1) Wir definieren $T' := (S', X, Y; R', s_0)$; dabei sei $Y' := Y \cup \{\beta\}$, β nicht $\in Y$;

R' enthalte die Regeln folgender Form:

a) $(s, p, q, t) \in R$ mit $l(p) \leq 1$, d.h. $p \in X$ oder $p = e$.

b) Seien für jede Regel $r = (s, p, q, t) \in R$, wobei

$p = x_1 \dots x_{k(r)}$ ein Eingabewort der Länge $k(r)$ sei,

$t_1^r, \dots, t_{k(r)-1}^r$ Symbole, die nicht in S enthalten

sind. Dann sind $(s, x_1, \beta, t_1^r), \dots, (t_{k(r)-2}^r, x_{k(r)-1}, \beta, t_{k(r)-1}^r)$,

$(t_{k(r)-1}^r, x_{k(r)}, q, t) \in R'$.

$S' := S \cup \{\text{alle } t_j^r\}$.

2) Nun definieren wir M und Q_{L_X} .

$M := \{ (t_1, w_1, z_1, t_2)(t_2, w_2, z_2, t_3) \dots (t_n, w_n, z_n, t_{n+1})$

$\in F(R') - \underline{e}$:

a) jedes $(t_i, w_i, z_i, t_{i+1}) \in R'$ ($1 \leq i \leq n$),

b) $t_1 = s_0$ und

c) $z_n \neq \beta$ }

$Q_{L_X} := \{ (t_1, w_1, z_1, t_2)(t_2, w_2, z_2, t_3) \dots (t_k, w_k, z_k, t_{k+1})$

$\in F(R') - \underline{e}$:

a) wie (a) von M ,

b) wie (b) von M und

c) $w_1 \dots w_k \in L_X$ }

3) Nun definieren wir eine reguläre Substitution f .

Sei für alle $(s, w, z, t) \in R'$:

$f((s, w, z, t)) := \begin{cases} \underline{z}, & \text{falls } z \neq \beta \\ \underline{e}, & \text{falls } z = \beta \end{cases}$

f ist offensichtlich eine reguläre Substitution. ^{+))}

^{+))} cf. Def. 6.2.1, p. 56 f.

4) Jetzt zeigen wir: $T_{s_0}^*(L_X) = f^+(M \cap Q_{L_X})$

Zunächst gilt:

- (x) $M \cap Q_{L_X} = \{ (t_1, w_1, z_1, t_2) (t_2, w_2, z_2, t_3) \dots (t_m, w_m, z_m, t_{m+1}) \in F(R') - e :$
 a) jedes $(t_i, w_i, z_i, t_{i+1}) \in R' \ (1 \leq i \leq m)$,
 b) $t_1 = s_0$,
 c) $z_m \neq \beta$ und
 d) $w_1 \dots w_m \in L_X \}$.

Man sieht nun sofort, daß $T_{s_0}^*(L_X) = f^+(M \cap Q_{L_X})$.

5) Als nächstes beweisen wir die Regularität von M. Wir zeigen: $F(R') - M \in RG(R')$, dann ist nach Satz 6.1.1 und dem AST auch $M \in RG(R')$.

$F(R') - M$ ist die Vereinigung von e und allen Mengen der Form

- a) $\underline{(s, w, z, t)} (R')^*$, wobei $(s, w, z, t) \in R'$ und $s \neq s_0$,
 b) $(R')^* \underline{(s, w, z, t)} \underline{(t', w', z', t'')} (R')^*$, wobei $(s, w, z, t), (t', w', z', t'') \in R'$ und $t \neq t'$
 oder
 c) $(R')^* \underline{(s, w, \beta, t)}$.

Mengen von Typ (a), (b) oder (c) sind offensichtlich in $RG(R')$. Wegen der Endlichkeit von R' gibt es ferner nur endlich viele solcher Mengen. Die endliche Vereinigung dieser Mengen ist gleich $F(R') - M$ und liegt wieder in $RG(R')$. Dies wollten wir zeigen.

6) Schließlich zeigen wir: $M \cap Q_{L_X} \in RG(R')$.

Zunächst definieren wir eine offensichtlich reguläre Substitution g. Sei für alle $x \in X$:

$$g(x) := \{ \text{alle } (s, e, z, t) \in R' \}^* \{ \text{alle } (s, x, z, t) \in R' \} \{ \text{alle } (s, e, z, t) \in R' \}^*.$$

Wir zeigen nun:

1. Falls e nicht $\in L_X$, so $M \cap Q_{L_X} = M \cap g^+(L_X)$.
 2) Falls $e \in L_X$, so $M \cap Q_{L_X} = M \cap (g^+(L_X) \cup \{ \text{alle } (s, e, z, t) \in R' \}^*)$.

In beiden Fällen ist $M \cap Q_{L_X}$ dann offensichtlich in $RG(R')$.

Zunächst gilt offenbar $Q_{L_X} \cap g^+(L_X)$. Daraus folgt für beide

Fälle die Inklusion von links nach rechts.

Nun zeigen wir die Inklusion von rechts nach links für den ersten Fall. Man mache sich klar, daß sich jeder Satz $p_1 \dots p_k$ von L_X in der Form $ep_1e \dots ep_ke$ schreiben läßt. Man sieht dann sofort ein, daß in $g^+(L_X)$ mindestens alle $(t_1, p_1, z_1, t_2) \dots (t_k, p_k, z_k, t_{k+1}) \in F(R')$ liegen mit $p_1 \dots p_k \in L_X$, d.h. alle Elemente von $g^+(L_X)$ genügen der Bedingung (d) von (x) auf p.60. In M liegen alle Worte aus $F(R') - e$, die den Bedingungen (a), (b) und (c) von (x) genügen. Mithin liegen in M und $g^+(L)$ alle Worte aus $F(R') - e$, die den Bedingungen (a) bis (d) von (x) genügen, also $M \cap g^+(L) \subseteq M \cap Q_{L_X}$. Die Inklusion von rechts nach links für den zweiten Fall zeigt man analog.

Damit ist Satz 6.3.1 bewiesen.

N.B. Satz 6.2.1 ist schwächer als Satz 6.3.1, denn jede reguläre Substitution kann als Sequenzentransducer-Abbildung dargestellt werden, nicht aber umgekehrt.⁺⁾ Neben seiner theoretischen Bedeutung ist dieser Satz aber auch für die Anwendung besonders wichtig. Er liefert die allgemeinste theoretische Grundlage für alle (dem Verfasser bekannten) Manipulationen auf der phonetischen, phonematischen und analogen "niedrigen" Ebenen der Sprachbeschreibung. Alle Arten der sogenannten "Finite-Transducers" gehören der Klasse der Sequenzentransducer an.

6.4 Ein Entscheidungsproblem

Die Klasse der FSSen läßt sich auch dadurch charakterisieren, daß für FSSen sämtliche Entscheidungsprobleme lösbar sind, die man im Zusammenhang mit Sprachen zu formulieren pflegt. Eine Übersicht über Entscheidungsprobleme, die generative Grammatiken betreffen, gibt LANDWEBER.⁺⁺⁾

In diesem Abschnitt wollen wir ein Beispiel für die Lösung eines Entscheidungsproblems geben, indem wir zeigen, daß es entscheidbar ist, ob zwei FSSen gleich sind. Unser Ziel ist also der Beweis von

Satz 6.4.1 Seien A und B DFAen mit m bzw. n Zuständen; sei $V_A = V_B$. Wenn $L(A)$ mit $L(B)$ für alle Ketten der Länge $m+n-1$ übereinstimmt, so $L(A) = L(B)$.

^{+) cf.} GINSBURG 1966, p.94

^{++) cf.} LANDWEBER, p.355

Weil es nur endlich viele Ketten der Länge $m+n-1$ gibt, folgt aus Satz 6.4.1 die Existenz eines Entscheidungsverfahrens für die Gleichheit von $L(A)$ und $L(B)$.

Sei für das folgende P_1 wie in Def.4.2.6 ⁺⁾ definiert; ferner bezeichne $|x|$ die Anzahl der Elemente der Menge x . Wir benötigen zum Beweis von Satz 6.4.1 nun die folgenden Hilfssätze. Dazu sei $A=(S,X,Y,n,o)$ ein bel. endlicher Automat.

HS.6.4.1 Wenn $|P_1|=1$, so $P_2=P_1$.

HS.6.4.2 Wenn $n=|S| \geq 2$, so $P_{n-1}=P_n$.

HS.6.4.3 Wenn $P_{k+1}=P_k$, so $s \text{ äq } t$ g.d.w. $s \text{ k-äq } t$ für bel. $s, t \in S$.

HS.6.4.4 Wenn $|S|=n$, so sind zwei Zustände von A äquivalent g.d.w. $O(s,p)=O(t,p)$ für alle p der Länge $n-1$.

HS.6.4.5 Seien A, B Automaten mit m bzw. n Zuständen. Sei $X_A=X_B$. Dann sind Zustände s von A und t von B äquivalent g.d.w. $M_A(s,p)=M_B(t,p)$ für jedes p der Länge $m+n-1$.

Aus HS.6.4.5 folgt sofort Satz 6.4.1. Seien nämlich $A=(S_A, X, n_A, s_0^A, F_A)$, $B=(S_B, X, n_B, s_0^B, F_B)$ DFAen mit m bzw. n Zuständen. Seien $A'=(S_A, X, \{0,1\}, n_A, m_A)$, $B'=(S_B, X, \{0,1\}, n_B, m_B)$ die durch A bzw. B eindeutig festgelegten MOORE-Automaten. ⁺⁺⁺⁾ Nach HS.6.4.5 gilt dann: $s_0^A \text{ äq } s_0^B$ g.d.w. $O_A(s_0^A, p)=O_B(s_0^B, p)$ für alle $p \in F(X)$ der Länge $m+n-1$. Offenbar sind s_0^A und s_0^B dann auch MOORE-äquivalent und folglich $L(A)=L(B)$. ⁺⁺⁾

Man stellt die Äquivalenz von DFAen hier also mit Hilfe der durch sie festgelegten MOORE-Automaten fest.

Bew. von HS.6.4.1: Sei $|P_1|=1$. Dann ist jeder Zustand von A 1-äq zu jedem anderen Zustand. Folglich gilt für alle $s, t \in S$, alle $x \in X$: $n(s, x) \text{ 1-äq } n(t, x)$. Folglich ist nach Lemma 4.2.1 ⁺⁾ $s \text{ 2-äq } t$. Also $P_1=P_2$.

⁺⁾ cf. p.38

⁺⁺⁾ cf. Satz 4.2.1, p.37

⁺⁺⁺⁾ cf. p.29

Bew. von HS.6.4.2: Sei $n=|S| \geq 2$.

1.Fall: $|P_1|=1$; dann ist nach HS.6.4.1 $P_1=P_2$ und nach Lemma 4.2.2 ⁺⁾ dann $P_2=P_{n-1}=P_n$.

2.Fall: $|P_1| \geq 2$. Aus Lemma 4.2.1 folgt, daß $|P_i| \leq |P_{i+1}|$ für alle i , denn $i+1$ -äq ist eine feinere Relation als i -äq. Weil jede Äquivalenzklasse von P_i mindestens einen Zustand enthält, so ist $|P_i| \leq n$ für alle i .

2.2.Fall: $|P_{n-1}|=n$, dann ist auch $|P_n|=n$, also $P_{n-1}=P_n$.

2.2.Fall: $|P_{n-1}| < n$, dann gibt es ein $k \leq n-2$ mit $|P_k|=|P_{k+1}|$, denn $|P_1| \leq 2$ und für jedes i ist P_{i+1} eine feinere Klasseneinteilung als P_i , d.h. wenn für alle $i, 1 \leq i < n-1$ $|P_{i+1}|$ größer als $|P_i|$ wäre, dann müßte $|P_{n-1}|=n$ sein, was nicht sein darf. Also ist auch $P_k=P_{k+1}$ und nach Lemma 4.2.2 $P_k=P_{n-1}=P_n$.

Bew. von HS.6.4.3: Sei $P_k=P_{k+1}$. Wenn $s \text{ äq } t$, so $s \text{ k-äq } t$. Sei $s \text{ k-äq } t$; nach Lemma 4.2.2 ist $s \text{ j-äq } t$ für alle $j \geq k$; also $s \text{ äq } t$.

Bew. von HS.6.4.4: O.E.d.A. können wir voraussetzen, daß $n \geq 2$. Nach HS.6.4.2 ist dann $P_{n-1}=P_n$. Nach HS.6.4.3 gilt dann: $s \text{ äq } t$ g.d.w. $s \text{ n-1-äq } t$.

Bew. von H.S.6.4.5: O.E.d.A. können wir voraussetzen, daß $S_A \cap S_B = \emptyset$. Sei $C=(S_A \cup S_B, X, Y, n_C, o_C)$ der Automat, dessen (n, o) -Matrix man erhält, indem man die (n, o) -Matrix von B direkt unter die von A schreibt. C hat $m+n$ Zustände und eine (n, o) -Matrix von $m+n$ Zeilen. Offensichtlich ist s von A äquivalent s von C und t von B äquivalent t von C für bel. s von A, t von B. Nach HS.6.4.4 gilt dann: $s \text{ äq } t$ g.d.w. $s \text{ m+n-1-äq } t$ für bel. s, t von C. Also gilt auch für bel. s von A, t von B: $s \text{ äq } t$ g.d.w. $O_A(s, p) = O(t, p)$ für alle p der Länge $m+n-1$.

N.B. Das hier angegebene Entscheidungsverfahren ist für praktische Zwecke nicht brauchbar. Der Grammatik, die im Anwendungsteil beschrieben ist, würde ein Automat mit mehreren tausend Zuständen entsprechen. Die Anzahl der zu testenden Ketten, um die Äquivalenz dieses Automaten mit einem anderen festzustellen, wäre unvorstellbar groß. ⁺⁺⁾

⁺⁾ cf. p.38 ⁺⁺⁾ cf. 3.5

Nachbemerkung zu Satz 6.2.1:⁺) Satz 6.2.1 ist für die Anwendung sehr interessant. Er liefert die theoretische Grundlage für ein Verfahren, das es erlaubt, Grammatiken in beliebig abgekürzter Notation zu schreiben. Man kann nämlich Grammatiken (DFAen) über einem Vokabular schreiben, dessen Elemente als Sprachen interpretiert werden können. Dieses Verfahren läßt sich beliebig oft iterieren und erlaubt so Darstellungen von großer Übersichtlichkeit. Wir werden auf diesen Punkt im Anwendungskapitel eingehen. ⁺⁺⁾

8. UMSCHREIBESYSTEME

Im folgenden untersuchen wir die Beziehungen der FSSen zu den durch Umschreibesysteme (Ersetzungssysteme, rewriting systems, Phrasenstrukturgrammatiken) beschreibbaren Sprachen. Derartige Beschreibungssysteme sind in der Literatur auch als CHOMSKY-Grammatiken bekannt. Von allen formalen Systemen sind sie bisher in der Linguistik am meisten benutzt und untersucht worden, so daß der Begriff 'CHOMSKY-Grammatik' meist synonym zu dem Begriff 'generative Grammatik' benutzt wird. ⁺⁺⁺⁾

Wie bei den TRn und Akzeptoren⁺⁺⁺⁺⁾ kann man auch bei den Umschreibesystemen verschiedene Typen unterscheiden. Zu einem bestimmten Grammatiktyp sind jeweils Grammatiken zusammengefaßt, die ein bestimmtes formales entscheidbares Merkmal besitzen. Die Menge aller Sprachen, die durch Grammatiken eines bestimmten Typs erzeugbar sind, nennen wir den diesem Grammatiktyp entsprechenden Sprachtyp. Verschiedenen CHOMSKY-Grammatiktypen entsprechen so verschiedene Sprachtypen, und man kann die Inklusionsverhältnisse dieser Sprachtypen zueinander untersuchen. Derartige Untersuchungen geschehen meist unter dem Stichwort CHOMSKY-Hierarchie. In dieser Hierarchie steht ein Grammatiktyp um so höher, je mehr Sprachen der ihm entsprechende Sprachtyp umfaßt. Einen in der CHOMSKY-Hierarchie höher

⁺) cf. p.57

⁺⁺⁾ cf. 8.1

⁺⁺⁺⁾ cf. 1.2 zum Terminus 'generative Grammatik'.

⁺⁺⁺⁺⁾ cf. p.8 f.

stehenden Grammatiktyp nennen wir auch generativ stärker als einen unter ihm stehenden Typ. Die CHOMSKY-Hierarchie gibt also Aufschluß über die generative Kraft von Grammatiktypen. Neben der Hierarchie von Grammatiktypen wollen wir auch von einer Hierarchie der ihnen entsprechenden Sprachtypen reden.

Da wir hier keine Untersuchungen über die CHOMSKY-Hierarchie durchführen wollen, sondern unser Ziel die Untersuchung der : zwischen CHOMSKY-Grammatiken und FAen bestehenden Verhältnisse ist, führen wir die Begriffe aus der Theorie der Umschreibesysteme nicht immer in voller Allgemeinheit ein, sondern nur so allgemein, wie wir sie für unsere Zwecke brauchen. Die Klasse der FSSen ist nämlich einem Sprachtyp extensional gleich, der in der CHOMSKY-Hierarchie sehr weit unten steht: unter ihm steht nur noch der Typ der endlichen Sprachen. Wir führen im folgenden die sogenannten kontextfreien Grammatiken(über V) und die einseitig-linearen Grammatiken (über V) ein. Ihnen entsprechen die Sprachtypen der kontextfreien Sprachen (über V) ($KF(V)$) bzw. der einseitig-linearen Sprachen (über V) ($EL(V)$). Wir zeigen dann die Gleichheit der Klassen $FS(V)$ und $EL(V)$. Ferner zeigen wir, daß $EL(V)$ echt in $KF(V)$ enthalten ist. Daraus folgt dann, daß $FS(V)$ echt in $KF(V)$ enthalten ist.

7.1 Grundlegende Begriffe der Theorie der kontextfreien Sprachen

In diesem Abschnitt führen wir die Begriffe ein, die wir benötigen, um die von uns intendierten Aussagen formulieren zu können.

Def.7.1.1 Eine kontextfreie Grammatik (über V) $G=(V,R,S)$ ist ein Tripel, wobei

- 1) V ein endliches Vokabular ist,
- 2) R eine Menge von Regeln der Form $A \rightarrow p$ ist, wobei $A \in V$, $p \in F(V)$,
- 3) S ein ausgezeichnetes Element von V ist.

Falls R keine Regel der Form $A \rightarrow e$ enthält, heißt G e-frei.

Def.7.1.2 Sei $G=(V,R,S)$ eine kontextfreie (kf) Grammatik;
dann sei

$A \in V_N$ g.d.w. 1) $A \in V$ und

2) es eine Regel $A \rightarrow p$ in R gibt.

V_N heißt Nonterminal-Vokabular von G , Elemente aus

V_N heißen Nonterminals oder Variablen.

$V_T = V - V_N$ heißt Terminal-Vokabular von G . Elemente aus V_T heißen Terminals oder Endsymbole.

Def.7.1.3 Eine Regel $A \rightarrow p$ ist anwendbar auf $w \in F(V)$ g.d.w.

es $z_1, z_2 \in F(V)$ gibt, so daß $w = z_1 A z_2$.

Das Resultat der Anwendung ist dann $z_1 p z_2$.

Eine Regel kann mehrfach anwendbar sein, und zwar dann, wenn es mehrere Paare z_1, z_2 im Sinne der Definition gibt. (Z.B. ist $A \rightarrow p$ auf $w = A x A$ auf zwei verschiedene Weisen anwendbar.)

Def.7.1.4 Aus p ist q direkt ableitbar: $p \rightarrow q$ g.d.w.

es eine Regel gibt, die auf p anwendbar ist und q das Resultat einer Anwendung dieser Regel ist.

Def.7.1.5 Aus p ist q ableitbar : $p \Rightarrow q$ g.d.w. $p = q$ oder es

$z_1, \dots, z_k$ gibt, so daß $z_1 \rightarrow z_2, z_2 \rightarrow z_3, \dots,$

$z_{k-1} \rightarrow z_k$ und $z_1 = p, z_k = q$. Abgekürzt schreiben wir

auch $z_1 \rightarrow z_2 \rightarrow \dots \rightarrow z_k$. Die Folge $z_1, \dots, z_k$ heißt Ableitung. Um in Zweifelsfällen klar zu machen,

daß sich eine Ableitung auf die Grammatik G bezieht, schreiben wir auch $p \xrightarrow{G} q$ bzw. $p \Rightarrow_G q$.

Man sieht sofort, daß $\Rightarrow$ eine reflexive und transitive Relation ist.

Def.7.1.6 Sei $G=(V,R,S)$ eine kf Grammatik. Dann heißt

$L(G) := \{ p \in F(V_T) : S \in p \}$ die von G erzeugte Sprache.

Sei $KF(V)$ die Menge aller durch kf Grammatiken erzeugbaren Sprachen. Jede Sprache in $KF(V)$ heißt kontextfrei (über V).

Def.7.1.7 Eine Grammatik $G=(S,R,V)$ ist rechts-linear (rl) (links-linear (ll)) (über V) g.d.w.

1) sie kf ist und

2) alle Regeln die Form $A \rightarrow p$ mit $p \in F(V_T)$ oder $A \rightarrow pB$ ($A \rightarrow \cdot p$) mit $p \in F(V_T), B \in V_N$ haben.

Sei $RL(V)$ bzw. $LL(V)$ die Klasse der durch rl bzw. ll Grammatiken (über V) erzeugbaren Sprachen.

Eine Grammatik heißt einseitig-linear (el), wenn sie rl oder ll ist. Sei $EL(V)$ die Klasse der el Sprachen

über V , d.h. der durch el Grammatiken über V erzeugbaren Sprachen.

Def.7.1.8 Eine Grammatik $G=(V,R,S)$ heißt Finite-State-Grammatik (FSG) g.d.w.

1) sie el ist und

2) alle Regeln die Form

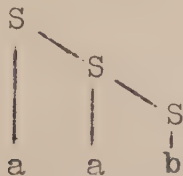
$$A \rightarrow x \text{ mit } x \in V_T \text{ oder}$$

$$A \rightarrow xB \text{ (} A \rightarrow Bx \text{) mit } x \in V_T, B \in V_N \text{ haben.}$$

Analog zur vorhergehenden Definition sprechen wir von rechts- bzw. links-linearen FSGen und rechts- bzw. linkslinaren FSSen. ⁺

Im Unterschied zu den FSSen, die durch FAen gegeben sind, werden also ll bzw. rl FSSen durch ll bzw. rl FSGen erzeugt. Wir benutzen diese an sich äquivalente Bezeichnung deshalb, weil sich später zeigen wird, daß rl FSGen und FAen praktisch dasselbe sind. ⁺)

Aus der Ableitung in einer FSG läßt sich sofort ein Baum ⁺⁺⁾ gewinnen. Man nehme etwa die FSG $G=(\{a,b,S\},R,S)$ mit $R=\{S \rightarrow aS, S \rightarrow b\}$. $S \rightarrow aS \rightarrow aaS \rightarrow aab$ ist dann offensichtlich eine Ableitung in G , der der folgende Baum entspricht:



Derartige sich stets nach rechts binär verzweigende Bäume sind typisch für rl Sprachen bzw. FSSen. Die Bäume von Sätzen ll Sprachen verzeigen sich analog nach links. Da derartige Bäume der Anschauung keine weitere Information liefern als z.B. eine Ableitung es tut, wollen wir sie im folgenden als einen Trivialfall von Baumstruktur beiseite lassen.

Def.7.1.9 Zwei Grammatiken G_1, G_2 heißen äquivalent g.d.w.
 $L(G_1)=L(G_2)$.

⁺) cf. den Bew. von Lemma 7.2.6

⁺⁺⁾ cf. p.6. Zur Def. des Begriffs Baum cf. GINSBURG 1966, p.23 ff.

7.2 Die Äquivalenz der Begriffe 'Finite-State-Akzeptor' und 'einseitig-lineare Grammatik'

Ebenso wie das AST hat das folgende Theorem die Äquivalenz zweier verschiedener Konzepte von formalen Systemen zum Inhalt.

Theorem 7.2.1 $FS(V) = EL(V)$

Zum Beweis dieses Theorems benötigen wir die folgende Definition und die folgenden Lemmata.

Def. 7.2.1 Sei $G = (V, R, S)$ eine kf Grammatik. Dann gelte für alle $A \in V_N$:

$$W_1(A) := \{A\},$$
$$W_{k+1}(A) := W_k \cup \{C \in V_N : \text{es gibt ein } B \in W_k : B \rightarrow C\}$$

für alle $k \geq 1$.

Lemma 7.2.1 Sei $G = (V, R, S)$ eine kf Grammatik; V_N enthalte n Elemente. Dann gilt für bel. $A, B \in V_N$:

$$A \Rightarrow B \text{ g.d.w. } B \in W_n(A).$$

Lemma 7.2.2 Sei $G = (V, R, S)$ eine rl Grammatik, $A, B \in V_N$, $A \neq B$ mit $A \Rightarrow B$ und $B \Rightarrow A$ (Dafür schreiben wir auch $A \Leftrightarrow B$). Sei $G' = (V - \{B\}, R', S)$, wobei R' genau aus den Regeln besteht, die man erhält, indem man jedes Vorkommen von B in jeder Regel aus R durch A ersetzt. Dann gilt: $L(G) = L(G')$.

Lemma 7.2.3 Zu jeder rl Grammatik G gibt es eine äquivalente rechtslineare Grammatik G' , die keine Regel der Form $A \rightarrow B$ enthält, wobei A und B Nonterminals von G' sind.

Lemma 7.2.4 Zu jeder rl Grammatik G gibt es eine äquivalente rl FSG G' . Falls G keine Regel der Form $A \rightarrow B$ hat, wobei A und B Nonterminals von G sind, so auch G' nicht.

Die Umkehrung von Lemma 7.2.4 gilt trivialerweise.

Lemma 7.2.5 $LL(V) = RL(V)$

Lemma 7.2.6 $RL(V) = FS(V)$

Aus Lemma 7.2.5 und Lemma 7.2.6 folgt sofort Theorem 7.2.1.

Lemma 7.2.1 wird für den Beweis von Lemma 7.2.2 benötigt, und die Lemmata 7.2.2 bis 7.2.5 brauchen wir, um Lemma 7.2.6 zu beweisen.

Beweis von Lemma 7.2.1: Sei $G=(V,R,S)$ ein kf Grammatik mit n Nonterminals; sei $A \in V_N$. Wir zeigen zunächst durch Induktion über k , daß für bel. $i \geq 1$ gilt:

(x) Wenn $W_i(A)=W_{i+1}(A)$, so $W_k(A)=W_i(A)$ für alle $k > i$.

I.A.: $k=i+1$: trivial

I.S.: Schluß von k auf $k+1$:

I.V.: $W_k(A)=W_i(A)$

I.B.: $W_{k+1}(A)=W_i(A)$

$$W_{k+1}(A)=W_k(A) \cup \{C \in V_N: \text{es gibt ein } B \in W_k(A): \\ B \rightarrow C\}$$

$$=W_i(A) \cup \{C \in V_N: \text{es gibt ein } B \in W_i(A): \\ B \rightarrow C\} \quad (\text{nach I.V.})$$

$$=W_{i+1}(A) \quad (\text{nach Def.})$$

$$=W_i(A) \quad (\text{nach Vor.}).$$

Damit ist (x) gezeigt.

Jedes $W_i(A)$ kann nun höchstens n Elemente enthalten, weil G nur n Nonterminals hat. $W_i(A)$ ist aber nur dann echt in W_{i+1} enthalten, wenn $W_{i+1}(A)$ mindestens ein Element mehr enthält als $W_i(A)$. Bei einer Grammatik mit n Nonterminals gilt dann auf jeden Fall $W_n(A)=W_{n+1}(A)$. Jedes $W_i(A)$ ist offenbar effektiv berechenbar. Wenn nun $A \Rightarrow B$, so gibt es eine Folge $A_1, \dots, A_r$ mit $A=A_1$, $B=A_r$ und $A_i \rightarrow A_{i+1}$ ($1 \leq i < r$). Jedes A_i ist offensichtlich eine Variable. Offenbar ist dann B auf jeden Fall in $W_r(A)$ und deshalb auch in $W_n(A)$. Wenn B in $W_n(A)$ liegt, dann nehme man das kleinste j , so daß $W_{n-j} \neq W_n(A)$, falls $W_n(A) \neq \{A\}$. Es gibt dann ein $A_{n-j} \in W_{n-j}(A)$, so daß $A_{n-j} \rightarrow B$. Dann gibt es auch $A_{n-(n-1)}, A_{n-(n-2)}, \dots, A_{n-(j-1)}, A_{n-j}$ mit $A_{n-(n-j)} \rightarrow \dots \rightarrow A_{n-j}$, wobei $A_{n-(n-j)}=A$. Also $A \Rightarrow B$. Falls $W_n = A$ und B in $W_n(A)$, so $A=B$ und folglich $A \Rightarrow B$.

Beweis von Lemma 7.2.2: Sei $G=(V,R,S)$ ein rl Grammatik, $A, B \in V_N$, $A \neq B$, $B \neq S$. Sei $G'=(V - \{B\}, R, S)$ die nach Konstruktion von Lemma 7.2.2 erzeugte Grammatik.

Wir zeigen: $L(G)=L(G')$. Sei $w_0 \rightarrow \dots \rightarrow w_k$ eine Ableitung in G mit $w_0=S$, $w_k \in F(V_T)$. In jedem w_i ($0 < i < k$) kommt als das am weitesten rechts stehende Symbol genau ein Nonterminal vor. Wir konstruieren eine Folge

$w_0', \dots, w_{k-1}', w_k'$ mit $w_0 = S$ und $w_k' = w_k$, indem wir jedes Vorkommen von B in der obigen Ableitung durch A ersetzen. Wir behaupten, daß $w_i' \rightarrow w_{i+1}'$ für alle $i, 0 \leq i < k$, daß mithin gilt: $S \Rightarrow w_k$. Falls nämlich $w_i \xrightarrow[G']{G} w_{i+1}$ und in w_i kein B vorkommt, so gilt: $w_i' \xrightarrow[G']{G} w_{i+1}'$. Falls $w_i \xrightarrow[G]{G} w_{i+1}$ und $w_i = pB$, so ist w_{i+1} das Resultat der Anwendung einer Regel $B \rightarrow qC$, also $w_{i+1} = pqC$. Nun ist nach Konstruktion $w_i' = pA$ und $w_{i+1}' = pqC$. Nach Konstruktion von G' gibt es aber eine Regel $A \rightarrow qC$ in G' , und daher ist w_{i+1}' das Resultat einer Anwendung dieser Regel. Damit ist die Inklusion von links nach rechts gezeigt.

Nun zeigen wir die andere Richtung der Inklusion. Sei $w_1, \dots, w_k$ eine Ableitung in G' mit $w_0 = S$ und $w_k \in F(V_T)$. Wir konstruieren eine Ableitung $S \Rightarrow w_k$ folgendermaßen: wir kontrollieren für alle $i, 0 \leq i < k$ ob $w_i \xrightarrow[G]{G} w_{i+1}$. Falls w_{i+1} nicht das Resultat einer Anwendung einer Regel von G auf w_i ist, so füge man das Standardzwischenstück $A_1, \dots, A_k$, das die Ableitung von B aus A ist, die es nach Vor. gibt, zwischen w_i und w_{i+1} ein. Wir erhalten so schließlich eine Folge $z_0, \dots, z_l$ mit $z_0 = S$ und $z_l = w_k$. Wir behaupten: $S \Rightarrow z_l$. Falls nämlich w_{i+1} aus w_i nicht in G ableitbar ist dann hat w_i die Form pA , und w_{i+1} hat die Form pqC , ist also Resultat einer Anwendung der Regel $A \rightarrow qC$. Nach Konstruktion gibt es dann eine Regel $B \rightarrow qC$ in G . Durch Einschoben des Standardzwischenstücks für $A \Rightarrow B : A_1, \dots, A_m$ zwischen w_i und w_{i+1} haben wir dann $pA \rightarrow pA_2 \rightarrow \dots \rightarrow pA_{m-1} \rightarrow pB \rightarrow pqC$ in G . Aus $w_i \xrightarrow[G']{G} w_{i+1}$ folgt also $w_i \xrightarrow[G]{G} w_{i+1}$. Damit gilt auch: $S \Rightarrow w_k = z_l$. q.e.d.

Beweis von Lemma 7.2.3: Sei $G = (V, R, S)$ eine rl Grammatik. Falls $L(G) = \emptyset$, so enthalte G' als einzige Regel $S \rightarrow pS$ (p ist ein bel. nichtleere Kette über V). Falls $L(G) = \underline{e}$, so enthalte G' als einzige Regel $S \rightarrow e$.

V_N enthalte n Nonterminals. Unter Anwendung der Lemmata 7.2.1 und 7.2.2 können wir eine äquivalente rl Grammatik konstruieren, die folgenden Einschränkungen genügt und die wir $G_1 = (V_1, R_1, S)$ nennen:

- 1) R_1 enthält keine Regeln der Form $A \rightarrow A$ mit $A \in V_N^1$; eine derartige Regel hat nämlich keinen Einfluß auf die generative Kraft einer Grammatik und kann daher weggelassen werden.
- 2) Falls A und B verschiedene Nonterminals von G_1 sind, so gilt: entweder $A \notin W_n(B)$ oder $B \notin W_n(A)$.

Nun definieren wir für alle $A, B \in V_N^1$:

$B \leq A$ g.d.w. $A=B$ oder es gibt eine Folge von Variablen

$A_1, \dots, A_k$, so daß $A_1=A$, $A_k=B$ und

$A_i \rightarrow A_{i+1} \in R_1$ für alle $i, 1 \leq i < k$. +)

Offenbar folgt aus $C \leq B$ und $B \leq A$, daß $C \leq A$. Ferner gilt wegen (2): wenn $B \leq A$ und $A \leq B$, so $A=B$. V_N^1 bildet also in bezug auf die Relation $\leq$ eine Halbordnung.

Seien $A, B \in V_N^1$ und sei $A \rightarrow B \in R_1$; sei C minimal bezüglich $\leq$, und es gelte: $C \leq A$. Wegen der Endlichkeit von V_N^1 gibt es ein solches C . Sei D eine Variable mit $D \rightarrow C$. Wegen (1) ist $C \neq D$. Seien $C \rightarrow w_1, \dots, C \rightarrow w_r$ alle Regeln in R_1 mit C links von $\rightarrow$. Weil C minimal bezüglich $\leq$ ist, ist kein w_i ($1 \leq i \leq r$) ein Nonterminal. Sei $R_2 := (R_1 - \{D \rightarrow C\}) \cup \{D \rightarrow w_i : 1 \leq i \leq r\}$.

Weil keines dieser w_i ein Nonterminal ist, enthält R_2 eine Regel der Form $A \rightarrow B$ weniger als R_1 . Sei $G_2 := (V_1, R_2, S)$. G_2 ist rl und es gilt: $L(G_1) = L(G_2)$. Sei nämlich $p_1, \dots, p_k$ eine Ableitung in G_1 . Dann ist entweder $p_1, \dots, p_k$ eine Ableitung in G_2 oder $p_{k-2} = zD$, $p_{k-1} = zC$ und $p_k = zw_i$ ($1 \leq i \leq r$). Wir ersetzen dieses Teilstück der Ableitung in G_1 durch die Folge zD, zw_i und erhalten so offenbar eine Ableitung in G_2 . Also $L(G_1) \subseteq L(G_2)$.

Sei $p_1, \dots, p_k$ eine Ableitung in G_2 . Falls $p_1, \dots, p_k$ keine Ableitung in G_1 ist, so $p_{k-1} = zD$, $p_k = zw_i$ ($1 \leq i \leq r$). Wir ersetzen dann p_{k-1}, p_k durch die Folge zD, zC, zw_i und erhalten so offensichtlich eine Ableitung in G_1 . Also $L(G_2) \subseteq L(G_1)$ und somit auch $L(G_1) = L(G_2)$.

Diese Konstruktion wird solange iteriert, bis wir schließlich eine Grammatik G' erhalten, die zu G äquivalent ist und keine Regel der Form $A \rightarrow B$ enthält.

+) Also $B \leq A$ g.d.w. $A \Rightarrow B$. Die Definition der Ableitbarkeitsbeziehung ist hier nur der Übersichtlichkeit noch einmal gebracht; cf. Def.7.1.5, p.66.

Beweis von Lemma 7.2.4: Sei $G=(V,R,S)$ eine rl Grammatik. Wegen Lemma 7.2.3 können o.E.d.A. annehmen, daß R keine Regel der Form $A \rightarrow B$ ($A, B \in V_N$) enthält. Nun konstruieren wir eine äquivalente rl FSG $G'=(V',R',S)$ wie folgt:
 R' enthalte die folgenden Regeln:

- 1) Wenn $r=A \rightarrow a_1 \dots a_k \in R$ mit $a_1, \dots, a_k \in V_T$,
 so $A \rightarrow a_1 A_r^1, A_r^i \rightarrow a_{i+1} A_r^{i+1} \in R'$ für alle $i, 1 \leq i \leq k-1$
 und $A_r^{k-1} \rightarrow a_k \in R'$.
- 2) Wenn $r=A \rightarrow a_1 \dots a_k B \in R$ mit $a_1, \dots, a_k \in V_T$ und $B \in V_N$,
 so $A \rightarrow a_1 A_r^1, A_r^i \rightarrow a_{i+1} A_r^{i+1} \in R'$ für alle $i, 1 \leq i \leq k-1$
 und $A_r^{k-1} \rightarrow a_k B \in R'$.

Die A_r^i 's sind nicht in V . Sei $V':=V \cup \{\text{alle } A_r^i\}$.

Ganz offensichtlich ist G' eine rl FSG und $L(G)=L(G')$.

Ferner enthält G' keine Regeln der Form $A \rightarrow B$ ($A, B \in V_N$) wenn G keine derartigen Regeln enthält.

Beweis von Lemma 7.2.5: Sei $G=(V,R,S)$ eine bel. rl Grammatik. Wir konstruieren eine äquivalente ll Grammatik $G'=(V,R',S)$. Daraus folgt dann $RL(V) \subseteq LL(V)$. Die Umkehrung ergibt sich aus Symmetriegründen. O.E.d.A. können wir wegen Lemma 7.2.3 voraussetzen, daß G keine Regeln der Form $A \rightarrow B$ mit $B \in V_N$ enthält.

G' hat folgende Regeln:

- 1) Wenn $A \rightarrow pB \in R$ ($B \in V_N$), so $\begin{cases} B \rightarrow Ap \in R', \text{ f. } A \neq S, \\ B \rightarrow p \in R', \text{ f. } A=S. \end{cases}$
- 2) Wenn $A \rightarrow p \in R$, so $\begin{cases} S \rightarrow Ap \in R', \text{ f. } A \neq S, \\ S \rightarrow p \in R', \text{ f. } A=S. \end{cases}$

Daß $L(G)=L(G')$ beweisen wir folgendermaßen (Die Fälle $L(G)=\emptyset$ und $L(G)=\underline{e}$ sind trivial und werden nicht berücksichtigt. Falls $e \in L(G)$, so muß G' natürlich auch eine Regel $S \rightarrow e$ haben.):

$p \in L(G)$ g.d.w. $p=p_1 \dots p_k$ ($p_i \in F(V)-\underline{e}$) und es in G entweder eine Ableitung

- a) $S \rightarrow p_1 A_1 \rightarrow p_1 p_2 A_2 \rightarrow \dots \rightarrow p_1 p_2 \dots p_{k-1} A_{k-1} \rightarrow p_1 \dots p_k$
 oder
- b) $S \rightarrow p_1 A_1 \rightarrow p_1 p_2 A_2 \rightarrow \dots, p_1 p_2 \dots p_k A_k \rightarrow p_1 \dots p_k$ gibt.

Im ersten Fall ist

$p_1 A_1$ Resultat der Anwendung von $S \rightarrow p_1 A_1$ auf S ,
 $p_1 p_2 A_2$ " " " " $A_1 \rightarrow p_2 A_2$ auf $p_1 A_1$,
 $\vdots$
 $p_1 \dots p_{k-1} A_{k-1}$ " " " " $A_{k-2} \rightarrow p_{k-1} A_{k-1}$ auf
 $p_1 \dots p_{k-2} A_{k-2}$,
 $p_1 \dots p_k$ " " " " $A_{k-1} \rightarrow p_k$ auf
 $p_1 \dots p_{k-1} A_k$:

Nach Konstruktion gibt es dann in R' Regeln

$S \rightarrow A_{k-1} p_k, A_{k-1} \rightarrow A_{k-2} p_{k-1}, \dots, A_1 \rightarrow p_1$ und folglich auch eine Ableitung für $p_1 \dots p_k$ in G' :

$S \rightarrow A_{k-1} p_k \rightarrow A_{k-2} p_{k-1} p_k \rightarrow \dots \rightarrow A_1 p_2 \dots p_k \rightarrow p_1 \dots p_k$.

Für den Fall (b) verfährt man ganz analog, nur ist

$p_1 \dots p_k$ dann Resultat einer Anwendung der Regel $A_k \rightarrow e$ auf $p_1 \dots p_k A_k$. Dann gibt es in R' Regeln $S \rightarrow A_k$,

$A_k \rightarrow A_{k-1} p_k, A_1 \rightarrow p_1$ ($A_k \rightarrow e$ wird in G' zu $S \rightarrow e A_k = S \rightarrow A_k$!) und folglich auch eine Ableitung für

$p_1 \dots p_k$ in G' . Also gilt: $L(G) \subseteq L(G')$. $L(G') \subseteq L(G)$ beweist man ganz analog.

Beweis von Lemma 7.2.6: Wir zeigen zuerst

Beh.1: $RL(V) \subseteq FS(V)$

Wir zeigen, daß es für eine bel. rl Grammatik G einen FA A gibt mit $L(G) = L(A)$. Wegen Theorem 4.1.1 ist dann Beh.1 bewiesen.

Sei $G = (V, R, S)$ eine bel. rl Grammatik. Wegen Lemma 7.2.3 können wir o.E.d.A. voraussetzen, daß R keine Regeln der Form $A \rightarrow B$ ($B \in V_N$) enthält. Nach Lemma

7.2.5 gibt es dann eine zu G äquivalente rl FSG

$G' = (V', R', S)$, die ebenfalls keine Regeln der Form

$A \rightarrow B$ ($B \in V_N$) enthält. Nun definieren wir den NFA

$A := (S_A, V_A, n, \{S\}, F)$ mit

$S_A := V_N \cup F$, wobei F ein Symbol t_A nicht $\in V_N$, enthalte und alle $A \in V_N$, für die gilt: $A \rightarrow e \in R'$; ferner sei $S \in F$ g.d.w. $e \in L(G')$.

$V_A := V_T \cup \dots$

Nun definieren wir für alle $A \in S_A, x \in V_A$:

$n(A, x) := \begin{cases} \{B \in V_N : \text{es gibt } A : A \rightarrow xB \in R'\}, & \text{f. } A \rightarrow x \in R' \\ \{B \in V_N : \text{es gibt } A : A \rightarrow xB \in R'\} \cup \{t_A\}, & \text{f. } A \rightarrow x \in R'. \end{cases}$

Man beachte, daß $n(t_A, x) = \emptyset$ für alle $x \in V_A$.

Nun gilt für ein bel. $p = x_1 \dots x_k$: $p \in L(G')$ g.d.w. es in G' entweder eine Ableitung $S \rightarrow x_1 A_1 \rightarrow \dots \rightarrow x_1 \dots x_{k-1} A_{k-1} \rightarrow x_1 \dots x_k$ gibt oder eine Ableitung in G' $S \rightarrow x_1 A_1 \rightarrow \dots \rightarrow x_1 \dots x_k A_k \rightarrow x_1 \dots x_k$ gibt. Dies ist genau dann der Fall, wenn $A_1 \in n(S, x_1)$, $A_2 \in n(A_1, x_2), \dots, A_{k-1} \in n(A_{k-2}, x_{k-1})$, $t_A \in n(A_{k-1}, x_k)$ bzw. $A_1 \in n(S, x_1), \dots, A_{k-1} \in n(A_{k-2}, x_{k-1})$, $A_k \in n(A_{k-1}, x_k)$ mit $A_k \in F$. In beiden Fällen ist also $p \in L(A)$. Also $L(G') = L(G) = L(A)$. Damit ist Beh.1 gezeigt.

Nun zeigen wir Beh.2: $FS(V) \subseteq RL(V)$.

Wir zeigen, daß es zu einem bel. DFA A eine rl Grammatik G gibt mit $L(A) = L(G)$.

Sei $A = (S_A, V_A, n, s_0, F)$ ein bel. DFA. O.E.d.A. können wir annehmen, daß $S_A \cap V_A = \emptyset$. Nun definieren wir die rl Grammatik $G := (V, R, s_0)$ mit

$V := S_A \cup V_A$ und

$R := \{ s \rightarrow xn(s, x) : s \in S_A, x \in V_A \} \cup \{ s \rightarrow e : s \in F \}$.

Nun zeigen wir: $L(A) \subseteq L(G)$

Sei p ein bel. Element von $L(A)$.

1. Fall: $p = e$, dann ist $s_0 \in F$ und folglich $s_0 \rightarrow e \in R$, also $e \in L(G)$.

2. Fall: $p = x_1 \dots x_k \in L(A)$ g.d.w.

$s_0 \rightarrow x_1 n(s_0, x_1)$

$\rightarrow x_1 x_2 n(n(s_0, x_1), x_2) = x_1 x_2 M(s_0, x_1 x_2)$

$\vdots$

$\rightarrow x_1 \dots x_k M(s_0, x_1 \dots x_k)$

$\rightarrow x_1 \dots x_k$

Also $L(A) \subseteq L(G)$.

Nun zeigen wir: $L(G) \subseteq L(A)$

Sei p ein bel. Element von $L(G)$.

1. Fall: $p = e$, dann ist $s_0 \rightarrow e \in R$, folglich $s_0 \in F$ und daher $e \in L(A)$.

2. Fall: $p \neq e$.

Wir betrachten zuerst eine bel. Ableitung in G : $s_0 \rightarrow p_1$

$\rightarrow \dots \rightarrow p_r$. In p_r komme dabei ein Nonterminal vor. Dieses ist das am weitesten rechts stehende Symbol der Kette p_r , weil G rechtslinear ist. Jedes $p_i \neq p_{i+1}$ ($1 \leq i < r$), denn G hat keine Regeln der Form $A \rightarrow A$. Unter diesen Bedingungen gilt also offenbar $p_i = x_1 \dots x_i M(s_0, x_1 \dots x_i)$ für alle

$i, 1 \leq i \leq r$. Sei nun $s_0 \rightarrow p_1 \rightarrow \dots \rightarrow p_k = p$ eine Ableitung in G und p_k enthalte kein Nonterminal. Dann muß p_{k-1} darstellbar sein als $p_k A$ und p_k das Resultat der Anwendung einer Regel $A \rightarrow e$ sein, denn G enthält nach Konstruktion keine Regel der Form $A \rightarrow x$ ($x \in V_T$). Nach dem Gesagten hat dann p_{k-1} die Form $p_k^M(s_0, p_k)$. Nach Konstruktion von G ist dann $M(s_0, p_k) \in F$, also $p \in L(A)$. Damit ist Beh.1 und folglich Lemma 7.2.6 bewiesen.

Die in diesem Beweis angegebenen Konstruktionen zeigen, daß man FSGen als FAen und umgekehrt auffassen kann. Man identifiziert einfach Nonterminals und Zustände und spielt die Regeln durch die Überföhrungsfunktion bzw. die Überföhrungsfunktion durch die Regeln nach. Man müßte alledings noch zulassen, daß eine FSG Regeln der Form $A \rightarrow e$ haben darf, damit die Entsprechung vollständig wäre. ⁺)

N.B. Für die Anwendung der Theorie ist besonders Lemma 7.2.3 wichtig. Wir benutzen später Grammatiken, die Regeln der Form $A \rightarrow B$ ($B \in V_N$) haben, denn derartige Grammatiken enthalten i.a. weniger Regeln als Grammatiken, die derartige Regeln nicht haben, und sie sind deshalb einfacher zu schreiben als jene. Wir weisen an dieser Stelle darauf hin, daß das theoretische Einfachheitskriterium der Minimalität einer Grammatik (nach dem in Analogie zu den DFAen eine minimale Grammatik eine mit möglichst wenigen Nonterminals unter allen zu ihr äquivalenten Grammatiken wäre) nicht unbedingt ein Kriterium dafür ist, ob eine Grammatik am einfachsten zu schreiben ist. Für praktische Zwecke könnte man eine Grammatik als optimal ansehen, wenn sie möglichst wenige Regeln hat (dies könnte z.B. für discovery procedures wichtig werden). CHOMSKY behauptet Lemma 7.2.3 analog für rechtslineare FSGen. Sein Beweis ist aber falsch, da er übersieht, daß Lemma 7.2.2 angewandt werden muß. ⁺⁺) Der Beweis von Theorem 7.2.1 findet sich in etwas anderer Form bei GINSBURG. ⁺⁺⁺)

⁺) cf. p.67 . Cf. (+) p.76

⁺⁺) cf. CHOMSKY 1958, p.97

⁺⁺⁺) cf. GINSBURG 1966, p.55 ff.

7.3 Folgerungen

In diesem Abschnitt wollen wir noch einige Folgerungen darstellen, die sich teils aus Theorem 7.2.1, teils aus früher bewiesenen Sätzen ergeben. Zuvor benötigen wir noch einige Definitionen:

Def.7.3.1 Sei G eine el Grammatik. Unter einer Struktur eines Satzes p von G verstehen wir die Folge der in einer Ableitung von p vorkommenden Nonterminals. Eine el Grammatik G ordnet jedem $p \in L(G)$ eine Menge von Strukturen $s_G(p)$ zu.

Def.7.3.2 Ein Satz p einer Sprache L ist in bezug auf eine rl Grammatik G von L strukturell eindeutig g.d.w. $s_G(p)$ genau ein Element enthält, strukturell mehrdeutig g.d.w. $s_G(p)$ mehr als ein Element enthält.

Def.7.3.3 Eine Grammatik G ist strukturell eindeutig g.d.w. jedes $p \in L(G)$ in bezug auf G strukturell eindeutig ist. Anderenfalls ist G strukturell mehrdeutig.

Offensichtlich gilt nun der folgende

Satz 7.3.1 Eine rl FSG $G=(V,R,S)$ ist strukturell eindeutig, wenn es zu jedem $A \in V_N$, $x \in V_T$ höchstens ein $B \in V_N$ gibt, so daß $A \rightarrow xB \in R$.

Ein DFA kann unter Berücksichtigung der p.74 gebrachten Konstruktion offenbar immer als strukturell eindeutige FSG aufgefaßt werden (die allerdings auch Regeln der Form $A \rightarrow e$ haben darf⁺). In diesem Sinne wollen wir deshalb auch von determinierten FSGen reden. Aber nicht jede strukturell eindeutige FSG ist determiniert, d.h. es gibt nicht für jedes $A \in V_N$, $x \in V_T$ höchstens ein B , so daß $A \rightarrow xB$ ($B \in V_N$). Man nehme etwa die rl FSG $G=(\{S,A,B\},R,S)$ mit $R=\{S \rightarrow aA, S \rightarrow aB, A \rightarrow b, B \rightarrow c\}$. Offenbar ist $L(G)=\{ab, ac\}$. ab und ac sind in bezug auf G strukturell eindeutig, aber die ersten beiden Regeln von G verletzen die Determiniertheitsbedingung, daß es zu jedem $A \in V_N$, $x \in V_T$ höchstens ein $B \in V_N$ mit $A \rightarrow xB \in R$ geben darf.

⁺) cf. p.75. Eine Regel der Form $A \rightarrow e$ muß gefordert werden um die Konstruktion von p.74 durchführen zu können.

Als Folgerung aus Satz 7.3.1 und den im Zusammenhang mit Theorem 7.2.1 bewiesenen Lemmata haben wir
Satz 7.3.2 Zu jeder el Grammatik G gibt es eine äquivalente FSG G' die strukturell eindeutig ist.

Beweis: Sei G eine bel. el Grammatik. O.E.d.A. können wir annehmen, daß G rl ist (wegen Lemma 7.2.5). Nach Lemma 7.2.6 gibt es dann einen NFA A mit $L(A)=L(G)$. Nach Theorem 4.1.1 gibt es einen DFA B mit $L(B)=L(A)$. Nach Lemma 7.2.6 gibt es dann zu B eine rl Grammatik G' mit $L(G')=L(B)=L(A)=L(G)$. G' ist offenbar sogar determiniert ⁺⁾ , also nach Satz 7.3.1 strukturell eindeutig .

Als Konsequenz aus Theorem 7.2.1 und Satz 4.3.1 ⁺⁺⁾ haben wir

Satz 7.3.3 $EL(V)$ ist echt in $KF(V)$ enthalten.

Beweis: In Satz 4.3.1 haben wir gesagt, daß die Sprache $L = \{ a^n b^n : n \geq 1 \}$ nicht in $FS(V)$ liegt. Wegen Theorem 7.2.1 liegt L dann auch nicht in $EL(V)$.

Andererseits liegt L aber in $KF(V)$. Dazu müssen wir nur eine kf Grammatik für L angeben. Sei

$G := (\{ S, a, b \}, R, S)$ mit $R := \{ S \rightarrow aSb, S \rightarrow ab \}$.

Offenbar ist G kf und $L(G)=L$.

Dieser Satz illustriert zugleich die Bemerkung im Anschluß von Def.4.3.1 ⁺⁺⁾, daß es durch Analysegrammatiken beschreibbare Sprachen gibt, die nicht durch DFAen beschreibbar sind. Jede kf Grammatik kann nämlich als Analysegrammatik aufgefaßt werden. ⁺⁺⁺⁾

Schließlich haben wir noch

Satz 7.3.4 Jede endliche Sprache ist eine FSS.

Beweis: Sei L eine bel. endliche Sprache, die aus den Sätzen $p_1, \dots, p_k \in F(V)$ bestehe. Wir können dann L durch eine Grammatik $G = (V \cup \{ S \}, R, S)$ (S nicht $\in V$) erzeugen, die die Regeln $S \rightarrow p_1, \dots, S \rightarrow p_k$ enthält. G ist el und wegen Theorem 7.2.1 gilt dann der Satz.

⁺⁾ cf. p.76

⁺⁺⁾ cf. p.41

⁺⁺⁺⁾ cf. GINSBURG 1966, p.116

DIE ANWENDUNG DER THEORIE AUF DAS ITALIENISCHE

Die Applikabilität der Theorie wurde mit Hilfe der Rechenanlage der Universität Münster untersucht. Als Programmiersprache wurde PL/I benutzt. Um nicht auf programmiertechnische Details eingehen zu müssen, was den Rahmen dieser Arbeit sprengen würde, begnügen wir uns damit, die Funktion und Leistungsfähigkeit der Programme zu erläutern. Die Programme sind nicht speziell auf das Italienische (It) zugeschnitten. Sie sind für die Bearbeitung jeder natürlichen Sprache, soweit dies mit Finite-State-Methoden möglich ist, brauchbar. Man kann sie so als zu dem alten Problemkreis der allgemeinen Grammatik zugehörig ansehen.

Wir geben zunächst eine Übersicht über die Programme. +)

1) 1) Das Explizitmachen einer abgekürzten FSG leistet das Programm EINLESEN. Eingegeben wird eine Rahmengrammatik und Regeln, die syntaktische Klassen definieren. Der Output besteht aus einer FSG über lexikalischen Kategorien des It..

2) Die Erzeugung von it. Sätzen leistet das Programm ITPG.

3) Die Analyse von it. Sätzen bzw. Nicht-Sätzen leistet das Programm ANAL, nachdem die zu analysierenden Eingabeketten über Wörtern des Schriftit. durch das Programm ITSU orthographisch präpariert und dann durch FTD; segmentiert worden sind.

4) Die Umwandlung einer unbeschränkten FSG in eine strukturell eindeutige FSG leistet die Programmfolge DPTILG, ETILG, EINDEUT (auf für den Zusammenhang unwichtige Zwischenprogramme wird im folgenden nicht eingegangen).

5) Die Reduktion (Minimierung) eines DFAs leistet das Programm MIN.

In den nun folgenden Abschnitten wollen wir diese Programme diskutieren.

8.1 Das Explizitmachen einer abgekürzten FSG

Die Regeln der hier realisierten FSG finden sich in der Anlage (dem Dokumentationsteil) p.22, Nr. 1175 - 1185.

In jeder Zeile steht ein 4-Tupel (Nonterminal, Eingabesymbol,

+) Ein Inhaltsverzeichnis der Programme findet sich im Dokumentationsteil.

Nonterminal, Wahrscheinlichkeitszahl). Es handelt sich um die Computer-nahe Notation für die Grammatik

$G_P := (V, R, 0)$

$R: 0 \rightarrow e \quad 1$

$0 \rightarrow \text{ADVBZ} \quad 1$

$\vdots$

$5 \rightarrow \text{ZEI}.$

(Die letzte Regel ist als $(5, \text{ZEI.}, 0, \text{Wahrscheinlichkeitszahl})$ realisiert, wobei 0 als Endzustand eines zu G_P äquivalenten FA angesehen werden kann. G_P kann nämlich offenbar auch sofort als DFA aufgefaßt werden. Wir verwenden i.f. daher auch die Automaten und Grammatikterminologie durcheinander, indem wir z.B. auch von Zuständen statt von Nonterminals etc. reden.)

$V_N := \{0, 1, \dots, 5\}$

$V_T := \{\text{ADVBZ}, \text{KAT-E}, \text{SK1}, \text{SK2}, \text{SK3}, \text{ZEI.}\}$

Wir wollen G_P Rahmengrammatik nennen. Wegen des AST können wir nun $L(G_P)$ äquivalent als reguläre Sprache darstellen, unter Beachtung der in 5.1 eingeführten Konventionen:

$((e \cup \text{ADVBZ})((\text{SK3 SK1} \cup \text{SK2})\text{KAT-E})^*(\text{SK3 SK1} \cup \text{SK2})) \cup (\text{SK3 SK1} \cup \text{SK2})(\text{ADVBZ} \cup e))\text{ZEI}.$

Führt man die in der linguistischen Literatur üblichen geschweiften Alternativklammern ein, so kann man übersichtlicher schreiben:

$$\left\{ \left\{ \begin{array}{l} e \\ \text{ADVBZ} \end{array} \right\} \left\{ \begin{array}{l} \text{SK3 SK1} \\ \text{SK2} \end{array} \right\} \text{KAT-E} \right\}^* \left\{ \begin{array}{l} \text{SK3 SK1} \\ \text{SK2} \end{array} \right\} \right\} \text{ZEI.}$$

$$\left\{ \left\{ \begin{array}{l} \text{SK3 SK1} \\ \text{SK2} \end{array} \right\} \left\{ \begin{array}{l} \text{ADVBZ} \\ e \end{array} \right\} \right\}$$

Die Rahmengrammatik kann man als Abkürzung für eine größere Grammatik auffassen. Man betrachte dazu etwa das Terminal SK1. Es bezeichnet die durch die Regeln Nr. 714 - 766 gegebene FSS (Anlage, p.13 ff.). Diese können wir für SK1 substituieren und erhalten dann wegen der Abgeschlossenheit der FSSen gegenüber der regulären Substitution (cf. Satz 6.2.1) wiederum eine FSS. In der Grammatik für die neue Sprache kommen eventuell wiederum substituierbare Symbole vor; man

kann also das Verfahren iterieren. Auf diese Weise läßt sich die harmlos anmutende Rahmengrammatik zu einer Grammatik von 8000 Regeln 'aufblasen'. Dies rekursive Verfahren leistet EINLESEN (cf. Anlage, p.1).

Das 'Aufblasen' legt nun folgende Redeweisen nahe: wir wollen syntaktische Kategorien die Symbole nennen, für die FSSen substituierbar sind; als lexikalische Kategorien wollen wir die Symbole bezeichnen, für die dies nicht der Fall ist. In den Regeln der 'voll aufgeblasenen' Grammatik kommen nur noch lexikalische Kategorien vor. Wegen ihres großen Umfanges ist die Grammatik im Anhang nicht ausgedruckt worden.

Die Technik des Aufblasens erlaubt so, Grammatiken in ökonomischer Weise zu schreiben und das Explizitmachen dem Programm EINLESEN zu überlassen. Allerdings haben gewisse sprachliche Phänomene ein rapides Anwachsen der Regelanzahl zur Folge. Z.B. hat das grammatisch recht durchsichtige Phänomen

{ADVBZ Satz}
{Satz ADVBZ} ,

das die Tatsache widerspiegelt, daß man sagen darf PRIMA EGLI PARTE und EGLI PARTE PRIMA, nicht aber * PRIMA EGLI PARTE PRIMA, eine Verdopplung der Anzahl der Regeln zur Folge. In unserem Fall handelt es sich um mehrere tausend Regeln!

8.2 Die Generation italienischer Sätze

Die Generation von italienischen Sätzen leistet ITPG. Wir erläutern das Funktionieren des Programms anhand des Beispiels LO SCIAMINO CHE VOI PORTASTE AMAVA L' ARANCIA CON EMOZIONE SCORAGGIANTE E SCIOCCA (Das Schwärmchen (d.h. der kleine Schwarm), das ihr trugt, liebte die Apfelsine mit entmutigender und törichter Erregung.).

Dieser Satz ist mit Hilfe der aufgeblasenen Rahmengrammatik, die wir der Einfachheit halber weiter G_p nennen wollen, erzeugt worden (cf. für das Folgende p.54 der Anlage). Die Regelwahl wurde durch einen Zufallsgenerator gesteuert.

Die Erzeugung können wir folgendermaßen beschreiben.

Zuerst generiert G_p die Kette von lexikalischen Kategorien

(x) DTISM SB1EIM IN- ... (3.Spalte auf p.54, von oben nach unten zu lesen), der die Struktur (cf. Def.7.3.1, p.76)

0,603,606,... entspricht.

Aus der auf p. 54 der Anlage gegebenen Information kann man ferner sofort die Ableitung

0,

DTISM 603,

DTISM SB1EIM 606,

DTISM SB1EIM 608, (Anwendung der Regel 606 → 608)

DTISM SB1EIM IN- 607,

:

konstruieren. +)

Für den nächsten Schritt der Erzeugung benötigen wir das Lexikon (Anlage, p.44 ff.). Es enthält Eintragungen, die jeweils aus einem it. Lexem (1. Spalte) und der dazugehörigen lexikalischen Kategorie (2. Spalte) bestehen. Man substituiert nun für jede lexikalische Kategorie der Kette (x) ein Lexem dieser Kategorie und erhält so schließlich eine Kette über den it. Lexemen, und zwar

(xx) LO SCIAM- IN- 0 ... (cf. p.54, 3.Spalte).

Die Auswahl der Lexeme ist durch einen Zufallsgenerator gesteuert, falls es mehrere Lexeme einer Kategorie gibt. Die Benutzung des Lexikons kann offenbar wiederum als reguläre Substitution aufgefaßt werden. ++)

Auf (xx) wird nun ein ST angesetzt, der (xx) in einen orthographisch korrekten it. Satz übersetzt. Wegen der Abgeschlossenheit der FSSen gegenüber ST-Abbildungen sind die so erhältlichen orthographisch korrekten Sätze mit Finite-State-Methoden beschreibbar. +++)

Die Regeln des ST stehen auf p.40 der Anlage in Computer-naher Schreibweise.

G_P leistet auch eine Art 'semantischer Steuerung'. So etwa wird AMAVA L" ARANCIA durch EMOZIONE SCORAGGIANTE ergänzt und nicht etwa durch z.B. STAZIONE. EMOZIONE

+) cf. Def.7.1.4 und 7.1.5, p.66

++) cf. Def.6.2.1, p.56 f.

+++ cf. Satz 6.3.1, p.58

wird ferner mit SCORAGGIANTE und SCIOCCA kombiniert, also mit im intuitiven Sinne 'semantisch verträglichen' Adjektiven. Die Kombination EMOZIONE ROSSA wäre auch in G_P zulässig, aber sie ist weniger wahrscheinlich. Die Wahrscheinlichkeit möglicher Kombinationen wird durch Wahrscheinlichkeitszahlen (Spalte 4, p.54) bestimmt. Im intuitiven Sinn 'semantisch abweichende' Sätze sind in G_P weniger wahrscheinlich als im intuitiven Sinn 'semantisch normale' Sätze. (D.h., wenn man endlich viele Sätze erzeugt, kommen mehr 'semantisch normale' als 'semantisch abweichende' im Sinne der Konzeption von G_P vor.)

Eine Eigenart von G_P ist es, daß sie die sogenannte Konstruktionshomonymie, d.h. die strukturelle Mehrdeutigkeit von Sätzen ⁺), nicht zuläßt. Ein im intuitiven Sinne mehrdeutiger Satz wäre etwa ECCO, LA PORTA ! . Man könnte ihn u.a. übersetzen als Da ist die Tür ! oder Sieh da, er trägt sie ! . In G_P hätte dieser Satz dennoch nur eine Struktur. PORTA setzt sich nämlich aus den Segmenten PORT- und A zusammen. PORT- gehört der lexikalischen Kategorie VB2AA/SB1F an (interpretierbar als 'Verb der Klasse 2 mit Stammvokal A, mit AVERE konjugiert' ⁺⁺⁺)). Mit anderen Worten, wir benutzen Kategorien der Form K1/K2, die bei inhaltlicher Deutung besagen, daß ein Wort der Kategorie K1/K2 intuitiv der Kategorie K1 oder der Kategorie K2 angehört. Das Problem der Konstruktionshomonymie ist also hier kein Realitäts- sondern ein Beschreibungsproblem.

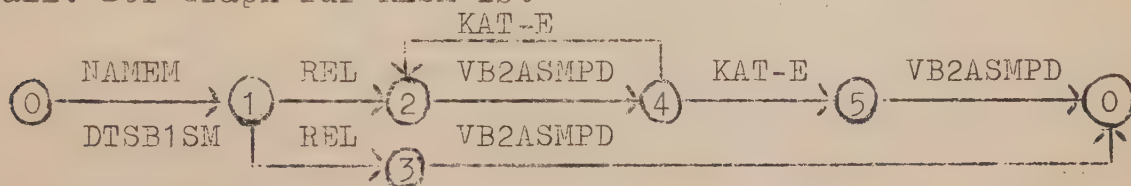
Nun noch einige Bemerkungen zur Heuristik. Eines der wichtigsten Probleme in diesem Zusammenhang ist die Fehlerkorrektur einer Grammatik. Die hier benutzte Methode scheint für diesen Zweck recht brauchbar zu sein. Beispielsweise ist der letzte Satz auf p.51 der Anlage IO PORTERO" LA VALIGIA STATICO... im intuitiven Sinne falsch; es muß heißen ESTATICA. Die Kategorie von STATICO ist ADIRVSM2; wir vermuten nun, daß der Fehler in dem durch ADSF1 bezeichneten Teilsystem vorkommt, denn ADSF1 ist die syntaktische Kategorie, für die weibliche Adjektive substituiert werden, und in den entsprechenden Regeln könnte fälschlicherweise

⁺) cf. Def.7.3.2, p.76 ⁺⁺⁺) 'oder Substantiv der Klasse 1 fém.' einfügen!

⁺⁺) Auf die inhaltliche Interpretation der Kategorien wird i.f. nicht näher eingegangen, da sie für das Verständnis der Methode nicht notwendig ist.

ein männliches Adjektiv sein (cf. p.3 der Anlage). Das durch ADSF1 bezeichnete Teilsystem ist aber korrekt. Jetzt muß der Fehler in einer syntaktischen Klasse vorkommen, die Ketten enthält, in denen ADSF1 vorkommt. Diese Bedingung erfüllen nur die Klassen SK1, SK2 und SK3. Tatsächlich kommt der Fehler in den Regeln für SK1 vor, nämlich Regel 728 (cf. Anlage, p.14) (2,ADSM2,4) muß ersetzt werden durch (2,ADSF2,4), wenn man die Wahrscheinlichkeitszahl hier außer acht läßt.

Ein größerer Fehler liegt in einer unkorrekten Relativsatzschachtelung (cf. Anlage, p.52) vor. Statt FAUSTO CHE ABBIAMO ASPETTATO E PORTAVO ... muß es heißen FAUSTO CHE ABBIAMO ASPETTATO E CHE PORTAVO... . Dieser Fehler muß in RLISM sein, denn nur dort wird dieser Typ von Relativsätzen beschrieben. Dies ist auch der Fall. Der Graph für RLISM ist:



Tatsächlich muß aber der von (4) nach (2) gehende Pfeil von (4) nach (1) gehen. Entsprechend sind die Regeln von RLISM zu ändern. Die syntaktischen Kategorien RLSE, RLPE und RLPM behandeln ganz analoge Relativsatzkonstruktionen. Man wird sie deshalb ebenfalls überprüfen, denn es steht zu vermuten, daß sich in ihnen analoge Fehler finden.

Die hier benutzte Methode ermöglicht so ein schnelles Auffinden von Fehlern und deren Korrektur.

8.3 Die Analyse von italienischen Sätzen bzw. Nicht-Sätzen

Die Analyse leistet die Programmfolge ANALYSE (cf. p.94a ff. der Anlage). Zur Analyse benutzen wir die determinierte FSG G_A , dh. eine Grammatik, die man als einen DFA ansehen kann. ^{+) Sie ist mit Hilfe der in 8.4 beschrie-}

^{+) cf. p.76}

benen Programme aus einer äquivalenten unbeschränkten ⁺) FSG berechnet worden. G_A ist nicht äquivalent zu G_P , denn G_P enthält zu viele Regeln, als daß die Rechnung erfolgreich hätte durchgeführt werden können. Das Beseitigen der Nicht-determiniertheit einer FSG läßt die Regelanzahl stark anschwellen. Deshalb scheiterte die Berechnung der zu G_P äquivalenten determinierten FSG an der beschränkten Kapazität des Kernspeichers der hiesigen Rechenanlage. ⁺⁺⁾

ANALYSE funktioniert folgendermaßen. Die Eingabeketten werden durch den ST ITSU für die Analyse orthographisch präpariert. Z.B wird DEL in DELIL umgeschrieben (wieder wird Satz 6.3.1 benutzt, der besagt, daß die präparierte Sprache wieder eine FSS ist). FTD1 segmentiert die präparierten Ketten; FTD1 kann ebenfalls als ST aufgefaßt werden. ANAL substituiert für die Segmente (it. Lexeme) lexikalische Kategorien (Anwendung von Satz 6.2.1) und analysiert die so entstehenden Kategorienketten.

⁺) cf. 8.4

⁺⁺⁾ Im April 1968 wurde die Rechnung für eine halbsogroße Grammatik erfolgreich durchgeführt. Die Eingabegrammatik enthielt 4013 Regeln, die berechnete determinierte Ausgabegrammatik 10218 Regeln. Das Protokoll der Rechnung kann wegen seines Umfanges im Dokumentationsteil nicht wiedergegeben werden. Es befindet sich in der Bibliothek des Fachbereichs Sprachwissenschaften der Universität Konstanz.
An dieser Stelle sei Herrn cand.math. Hartmut MEIER vom Rechenzentrum der Universität Münster herzlich gedankt. Er hat die Datenorganisation für die Rechnung besorgt und in selbstloser Weise mehrere Nächte für die Rechnung geopfert, die ohne seine Hilfe nicht hätte durchgeführt werden können.

Man veranschauliche sich die 'Genauigkeit' der Analyse an folgenden Beispielen (cf. dazu ANALYSE, p.156 der Anlage). Eingabekette 1 IL BAULONINO E" BELLO . ist im intuitiven Sinne korrekt (man kann jedenfalls der Meinung sein), auf Grund der Regeln von G_A aber ungrammatisch. Eingabekette 4 unterscheidet sich von Eingabekette 3 nur durch den fehlenden Akzent auf dem A von COMINCERA" (der hier durch ' " ' dargestellt ist); 4 wird nicht akzeptiert. In Eingabekett 10 fehlt ein Komma, und 11 ist Nonsens. Das Programm erkennt alle diese Fälle.

8.4 Die Umwandlung einer unbeschränkten FSG in eine determinierte FSG

Eingegeben wird eine rl Grammatik $G=(V,R,O)$, deren Regeln sämtlich die Form $A \rightarrow x$ mit $x \in V_T$ (realisiert als $(A,x,O)^{+})$, $A \rightarrow xB$ ($B \in V_N$, $x \in V_T$) oder $A \rightarrow B$ ($B \in V_N$) haben. Eine Grammatik dieses Typs wollen wir unbeschränkte FSG nennen.⁺⁺)

DPTILG verwandelt G in eine äquivalente Grammatik G' , in der es keine Nonterminals A,B gibt mit $A \Leftrightarrow B$. Es handelt sich hier also um eine Realisierung von Lemma 7.2.2. ⁺⁺⁺)

ETILG realisiert Lemma 7.2.3, indem zu G' eine äquivalente FSG G'' ohne Regeln der Form $A \rightarrow B$ ($B \in V_N$) konstruiert wird. ⁺⁺⁺)

EINDEUT konstruiert eine zu G'' äquivalente determinierte (und daher auch strukturell eindeutige ⁺⁺⁺⁺) Grammatik G''' . G''' läßt sich als DFA auffassen. Wir haben also hier eine Realisierung von Theorem 4.1.1 vorliegen. ⁺⁺⁺⁺)

⁺) cf. p.79

⁺⁺) cf. CHOMSKY 1958, p.96

⁺⁺⁺) cf. p.68

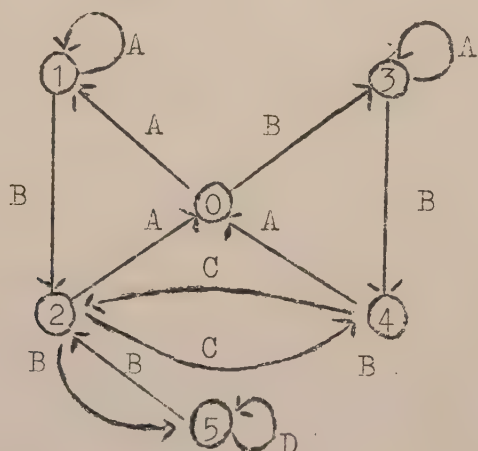
⁺⁺⁺⁺) cf. p.32

⁺⁺⁺⁺⁺) cf. Satz 7.3.1, p.76

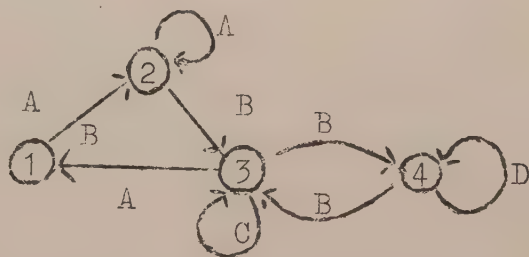
8.5. Ein Algorithmus zur MOORE-Reduktion von DFAen

MIN ist die Realisierung von Theorem 4.2.1 ⁺) (cf. für das Folgende p.74f. der Anlage). Das benutzte Verfahren ist auf p.39 f. beschrieben. Der Output von MIN (cf. p.90 der Anlage) bringt die i-Tabellen ($i=1,2,3,4$). Die Zeilen der Tabellen entsprechen den Zuständen 0 bis 6, die Spalten den Symbolen A,B,C,D. In den i-Tabellen sind die i-Klassen eingetragen. Z.B. stehen in der 0-Tabelle die 0-Klassen 7 und 8 wobei $7 = \{0\}$ und $8 = \{1,2,3,4,5,6\}$. Die Elemente der i-Klassen lassen sich aus dem Output, der jeweils unter den i-Tabellen steht, leicht ablesen. Z.B. bedeutet 86123456 (das Ganze senkrecht geschrieben!), daß die Klasse 8 6 Elemente enthält, nämlich die Zustände 1 bis 6. 710 bedeutet entsprechend, daß 7 nur das Element 0 enthält.

MIN verwandelt nun den DFA



in den MOORE-reduzierten DFA



Anfangs- und Endzustand im alten DFA ist 0, im neuen 1.

⁺) cf. p.39

8.6 Zusammenfassung der Ergebnisse

Die Theorie der FSSen hat schöne theoretische Eigenschaften: die üblichen Entscheidungsprobleme sind lösbar, interessante Abgeschlossenheitseigenschaften gelten, und verschiedene traditionelle Begriffsbildungen sind gut explizierbar.

Gegen diese Theorie ist bisher vor allem eingewandt worden, daß sich nicht alle Phänomene von natürlichen Sprachen mit ihren Methoden beschreiben lassen. Dafür gibt etwa CHOMSKY Beispiele.⁺⁾ Uns scheint dies nicht der entscheidende Einwand zu sein, sondern dieser - so scheint uns - muß sich gegen die Schwerfälligkeit der Handhabung der Regeln richten. Dies äußert sich in der gewaltigen Größe von Grammatiken mit bescheidener Leistung.

Wie gefährlich es ist, Aussagen über die Vorzüge und Nachteile einer Theorie ohne experimentelle Untersuchungen zu machen - besonders wenn es um die Anwendbarkeit einer Theorie geht - scheint uns die folgende Bemerkung von BAR-HILLEL zu illustrieren⁺⁺⁺⁾: 'The use of a FSG model has the following advantage: Suppose two (or more) grammarians suggest different grammars for the same language. If these grammars are FSGs, we have a mechanical method of checking whether they are equivalent or not.' Nehmen wir an, wir hätten eine Grammatik mit 500 Lexemen; bereits unsere Grammatik G_p hat etwa 2500 Zustände. Für eine vergleichbare Grammatik mögen dieselben Größenverhältnisse gelten. Um die Äquivalenz beider Grammatiken mechanisch zu testen, müßten wir alle Lexemketten etwa der Länge 5000 vergleichen.⁺⁺⁾ Diese Zahl, deren Größe jede Anschauung übersteigt, zeigt, daß mit der bloßen Angabe eines mechanischen Verfahrens zur Lösung eines Problems für die Anwendung einer Theorie noch nichts gewonnen sein muß.

^{+) cf. CHOMSKY 1957, p.22}

^{++) cf. 6.4, p.61 ff.}

⁺⁺⁺⁾ cf. BAR-HILLEL, p.94

Die Bedeutung der Theorie der FSSen für die 'niedrigen' Ebenen der Sprachbeschreibung scheint dem Verfasser jedoch gewiß zu sein. Die Kombination von Transducern mit Grammatiken dürfte auch für spätere Anwendungen wichtig sein. Ferner könnte der Theorie für die Beschreibung der Lautgesetze, also für die historische Grammatik, wichtig werden. Dies erscheint dem Verfasser plausibel, müßte aber noch näher begründet werden.

ZUR BENUTZTEN LITERATUR

1. Yehoshua BAR-HILLEL (BAR-HILLEL): Finite-State Languages: Formal Representations and Adequacy Problems; in : Language and Information, Jerusalem 1964.
2. Noam CHOMSKY (CHOMSKY 1957): Syntactic Structures; s'Gravenhage 1957.
3. Noam CHOMSKY and George A. MILLER (CHOMSKY 1958): Finite State Languages; in Information and Control 1, 91-112 (1958).
4. Seymour GINSBURG (GINSBURG 1962): An Introduction to Mathematical Machine Theory; Reading (Mass.) 1962.
5. Seymour GINSBURG (GINSBURG 1966): The Mathematical Theory of Context-Free Languages; New York 1966.
6. W.M. GLUSCHKOW (GLUSCHKOW): Theorie der abstrakten Automaten; Mathematische Forschungsberichte XIX, Berlin 1963.
7. Hans HERMES (HERMES): Aufzählbarkeit, Entscheidbarkeit, Berechenbarkeit; Berlin 1961.
8. Peter S. LANDWEBER (LANDWEBER): Decision Problems of Phrase-Structure Grammars; Institute of Electrical Engineers, Transactions on Electronic Computers, vol. EC 13 (1964); 354-63.
9. M.O. RABIN and D. SCOTT (RABIN-SCOTT): Finite Automata and Their Decision Problems; IBM Journal of Research and Development 3 (1959), 114-125.

Der im theoretischen Teil der Arbeit⁺ findet sich zum größten Teil in 4 und 5. Es wird hier nur solche Literatur angeführt, auf die ausdrücklich verwiesen wird. Die Begriffsbildungen in den Kapiteln 1 und 7 gehen z.T. auf Vorlesungen von Dr. Klaus BROCKHAUS (WS 1967/68, SS 1968) zurück, aus denen einige Formulierungen fast wörtlich übernommen wurden. Für wichtige Hinweise zur Korrektur der Arbeit bin ich Herrn cand.math. Robert MAIER verpflichtet.

⁺) 'gebrachte Stoff' einfügen!

SYMBOLE UND ABKÜRZUNGEN

$(A)^*$	Def.5.1.0, p.42
(AB)	Def.5.1.0, p.42
$(A \cup B)$	Def.5.1.0, p.42
$\dot{a}q$	Def.2.1.5, p.14
$A_s(F(X))$	Def.2.1.4, p.14
AST	p.42
DFA	Def.4.1.0, p.30
e	Def.1.2.2, p.3
el	Def.7.1.7, p.66
$EL(V)$	Def.7.1.7, p.66
$F(V)$	Def. 1.2.3, p.3
$FS(V)$	Def.4.1.3, p.41
FSG	Def.7.1.8, p.67
FSS	Def.4.1.3, p.41
G	Def.1.2.6 und Def.1.2.7, p.4
$k\text{-}\dot{a}q$	Def.4.2.6, p.38
kf	Def.7.1.2, p.66
$KF(V)$	Def.7.1.6, p.66
ll	Def.7.1.7, p.66
$L(A)$	Def.4.1.1, p.30 und Def.4.1.3, p.31
$L(G)$	Def.1.2.6, p.4 und Def.7.1.6, p.66
L_I^F	Def.1.3.7, p.9
$l(p)$	Def.1.2.2, p.3
L_V	Def.1.2.4, p.3
$m(s)$	Def.2.3.1, p.18
$M(s,p)$	Def.3.3.1, p.27
$MO\text{-}\dot{a}q$	Def. 2.3.3, p.20 (Automaten) und Def.4.2.3, p.36 (Finite-State-Akzeptoren)
$M_1 \boxtimes \dots \boxtimes M_n$	(++), p.6
$n(s,x)$	Def.2.1.1, p.10
$N(s,p)$	Def.2.1.1, p.10
NFA	Def.4.1.2, p.31
$o(s,x)$	Def.2.1.1, p.10
$O(s,x)$	Def.2.1.2, p.13
$P(M)$	(++), p.7
$\bar{p}$	Def.5.2.2, p.50
P_k	Def.4.2.6, p.38

$p \rightarrow q$	Def.7.1.4, p.66
$p \Rightarrow q$	Def.7.1.5, p.66
$p \Leftrightarrow q$	Lemma 7.2.2, p.68
$RG(V)$	Def.5.1.1, p.42
rl	Def.7.1.7, p.66
$s_G(p)$	Def.7.3.1, p.76
ST	Def.6.3.1, p.58
S_y	Def.5.2.5, p.25
TR	Def.1.3.1, p.6
$T_s(p)$	Def.1.3.2, p.7
$T_s^*(L)$	Def.1.3.2, p.7
V	Def.1.2.1, p.3
$/x/$	p.62

SACHVERZEICHNIS

- Abbildung, von Automat erzeugte: Def.2.1.4, p.14
ableitbar, Ableitung: Def.7.1.5, p.66
akzeptierende Zustandsfolge: Def.4.1.3, p.31
Akzeptor: Def.1.3.6, p.9
Alphabet: cf. Vokabular
Analysegrammatik: cf. Erkennungsgrammatik
äquivalent: Def.2.1.5, p.14
Äquivalenz: cf. äquivalent
Äquivalenz: Ä. von Finite-State-Akzeptoren, Def.4.1.4, p.31
 Ä. von Grammatiken, Def.7.1.9, p.67
Äquivalenzklasse: Def.5.2.2, p.50
aufzählbar: Def.1.2.6, p.4
Automat: cf. MEALY-Automat
 initialer A.: cf. p.10
Automatenabbildung: Def.3.2.3, p.24
CHOMSKY-Hierarchie: p.64
Darstellbarkeit von Sprachen durch (in) Automaten:
 Def.3.2.1, p.23
 D. von Abbildungen durch Automaten: Def.3.2.2, p.24
determiniert: Def.1.3.3, p.8
Disjunktion von Sprachen: Def.5.1.0, p.42
endlicher Index einer Äquivalenzrelation: Def.5.2.2, p.50

endlich: Def.1.3.5, p.8 (in bezug auf Transducer)
Endzustände eines Automaten: Def.3.3.2, p.28
entscheidbar: Def.1.2.7
Entscheidungsverfahren: Def.1.2.7, p.4
Erkennungsgrammatik: Def.1.2.7, p.4
Ersetzungssystem: cf. Umschreibesystem
Finite-State-Akzeptor: determinierter FSA: Def.4.1.0, p.30
 nichtdeterminierter FSA: Def.4.1.2, p.31
Finite-State-Grammatik: Def.7.1.8, p.67
 determinierte FSG: p.76
Finite-State-Sprache: Def.4.1.3, p.41
Grammatik: generative G.: Def.1.2.6, p.4
 kontextfreie G.: Def.7.1.1, p.65
 rechts- (links-) lineare G.: Def.7.1.7, p.66
Grammatiktyp: p.64
Halbgruppe, freie: cf. Monoid
Homomorphismus: Def.2.1.3, p.13 und Def.4.2.1, p.35
Isomorphismus: cf. Homomorphismus
Kategorie, syntaktische, lexikalische: p.80
Kette (über V): Def.1.2.2, p.3
k-Klassen: Def.4.2.6, p.39
Kongruenzrelation: Def.5.2.1, p.50
Länge (einer Kette): Def.1.2.2, p.3
Markierungsfunktion: Def.2.3.1, p.18
MEALY-Automat: Def.2.1.1, p.10
minimal: cf. reduziert
Monoid, freies: Def.1.2.3, p.3
MOORE-Äquivalenz: MO-Ä. von Automaten: Def.2.3.3, p.20
 MO-Ä. von Finite-State-Akzeptoren: Def.4.2.2 und Def.
 4.2.3, p.36
MOORE-Automat: Def.2.3.1, p.18
MOORE-Homomorphismus: Def.2.3.1, p.20
Nonterminal: Def.7.1.2, p.66
Output-Funktion: Def.2.1.1, p.10
 erweiterte O.-Fkt., p.13
Produkt von Sprachen: Def.5.1.0, p.42
reduziert: r. Automat: Def.2.2.1, p.18
 MOORE-reduzierter MOORE-Automat, Def.2.2.4, p.20
 r. determinierter Finite-State-Akzeptor, Def.4.2.4, p.37

reguläre Sprache: Def.5.1.1, p.42
Satz einer Sprache: Def.1.2.5, p.4
Sequenzentransducer: p.8 und Def.6.3.1, p.58
Sequenzentransducer-Abbildung: Def.6.3.2, p.58
Sprache: natürliche S.: p.4
 durch Zustandsmenge festgelegte S.: Def.1.3.7, p.9
 S.über V: Def.1.2.4, p.3
Sprachtyp: p.64
Sternprodukt einer Sprache : Def.5.1.0, p.42
Struktur eines Satzes: Def.7.3.1, p.76
strukturell eindeutig (mehrdeutig): Def.7.3.2 und
 Def.7.3.3, p.76
Substitution: Def.6.2.1, p.56
Terminal: Def.7.1.2, p.66
Transducer : Def.1.3.1, p.6
Transducer-Abbildung: Def.1.3.2, p.7
Übersetzung (des Satzes p): Def.1.3.2, p.7
Überföhrungsfunktion:eines Automaten: p.10
 erweiterte Ü.-Fkt. N: Def.2.1.2, p.13
 verallgemeinerte Ü.-Fkt. M: Def.3.3.1, p.27
Umschreibesystem: p.6
Variable: cf. Nonterminal
Vokabular: Def.1.2.1, p.3
zusammenhängend: Def.4.2.5, p.38

ERRATA

In Def.1.2.2, p.3 muß es heißen:'Ein Vokabular V ...'.
p.64 muß es heißen:'8. UMSCHREIBESYSTEME'.

LEBENS LAUF

Am 28.7.1941 wurde ich, Arnim von Stechow, als Sohn des jetzigen Botschafters i.R. Johann Karl von Stechow und seiner Frau Olga Maria, geb. Ostrowska, in Berlin geboren. 1948 bis 1952 besuchte ich die Volksschule in Reinbek bei Hamburg. 1952 bis 1960 war ich Schüler der folgenden Humanistischen Gymnasien: Istituto Giulia, Mailand (1952-55), Staatliches Nikolaus Cusanus Gymnasium, Bad Godesberg (1955-57) und Zinzendorf Gymnasium, Königsfeld/Schwarzwald (1957-60). Nach meinem Abitur im März 1960 leistete ich den Wehrdienst bis Weihnachten 1961 ab. Im Wintersemester 1961/62 immatrikulierte ich mich an der Philosophischen Fakultät der Universität Bonn für die Fächer Philosophie und Geschichte. Im Sommersemester 1962 begann ich, Romanistik zu studieren, und seit dem Wintersemester 1962/63 hörte ich sprachwissenschaftliche Vorlesungen. Im Wintersemester 1963/64 studierte ich in Grenoble. Es folgten zwei Semester in Münster; statt Geschichte studierte ich Sprachwissenschaft (1964/65). Während des Sommersemesters 1965 war ich beurlaubt und lernte Spanisch am Colegio San Juan de Letran, Manila (Philippinen). Seit dem Wintersemester 1965/66 studierte ich wieder in Münster und hörte neben Sprachwissenschaft, Romanischer Philologie und Philosophie auch mathematische Logik.

Meine akademischen Lehrer waren unter anderen die Damen und Herren Brockhaus, Coseriu, Greive, K. Hartmann, P. Hartmann, Harweg, Heidemann, Hermes, Kaulbach, Lausberg, Litt, Marquard, Martin, H.Meier, Nolting-Hauff, Ott, Perpeet, Ritter-Santini, Rödding, Specht und Wagner.

